The Best of
creative computing

Volume 3

Edited by
David H. Ahl
and
Burchenal Green

The Best of Creative Computing Volume 3

The Best of
creative computing
Volume 3

Edited by David H. Ahl
and
Burchenal Green

Creative Computing Press, Morris Plains, New Jersey

Copyright © 1980 by Creative Computing Press.

All rights reserved. No portion of this book may
be reproduced—mechanically, electronically or by any
other means, including photocopying—without written
permission of the publisher.

Library of Congress Number: 76-438v. 3
ISBN: 0-916688-12-7

Printed in the United States of America

10 9 8 7 6 5 4 3 2

Creative Computing Press
39 E. Hanover Ave.
Morris Plains, New Jersey 07950

Preface

This book is a collection of the best articles, activities, fiction, games, programs, reviews, cartoons and other information from the six 1977 issues of *Creative Computing*. It's been over two and a half years since the last issue of Volume 3 was put to bed. In these few short years the growth in computing accessibility to individuals has been phenomenal. Basically that's what *Creative Computing* is all about—helping people use computers in innovative ways.

People who want to use computers for art, music, education, simulation, modeling, budgeting, data base management, text editing, conferencing, or game playing will turn to *Creative Computing*. People who want to keep up to date on the role of the computer in society will be equally enthusiastic.

But material over two years old? Considering how the computer industry has grown, with the rapid development of new computers and peripherals, and the software capability only wished for in 1977, wouldn't this material have become outdated? Would not it at best provide an historical perspective?

Reviewing the 1977 issues to compile this work, I was amazed at how current and vital the contents still are. It is the nature of the material itself that dictates this. Most of the authors are still in the forefront of this new field, pioneers in the industry. Their work captures the dedication to growth, the excitement in discovery, the stimulation in sharing knowledge that creates memorable articles, thorough reviews, worthwhile activities and programs, and imaginative games. This book, then, proves to be more than a foundation from which the reader can build his knowledge. Much of the material in this volume doesn't yet have an 'update.'

A good game is fun. Well written problem solving and programming techniques, language and programming theory materials are still scarce. Properly conceived applications for education, art, music, and science can still be state-of-the-art.

Creative Computing, like the whole world of computing of which it is a part, seems to be perpetually changing—growing, improving, shifting gears. Yet, to a great extent, the themes remain constant. There are always features on artificial intelligence, art and graphics, education, medicine, text editing, music, and public access. More information. New information. Not a static repetition but a reverberation of shared knowledge that shapes new information. These themes were all found in the pages of Volume 3, and provide this collection with an impressive array of topics and authors.

The book is a chapter in the development of personal computing. But, because it will generate new ideas, work and programs that will appear in future *Creative Computing* issues, it is still an open chapter. It is 336 printed pages, but it is also a spirit of adventure and creation.

And growth. In many ways *Creative Computing* is a microcosm of the whole personal computing industry. David Ahl began *Creative Computing* as 68 pages of newsprint in a corner of his home office. His kids used to hand label the magazines for mailing. The day he got his one thousandth subscriber he had his grandest champagne celebration. But now that Creative has two buildings, fifty employees, a 192-page 'slick' magazine with a circulation of 80,000, there are no more gala celebrations. Everybody's too busy.

Burchenal Green
May 1980

Contents

Articles and Commentary

• Technology—Present and Future

2 Trends Into the Future .... Gray

3 EFTS: Living Is Better Electronically, or IS It? .... Dragunas

5 The World In Your Own Notebook .... Lees

8 Eeny, Meeny, Micro and More .... Salisbury

12 The Pocket Computer Is Almost Here .... Ahern

14 Microprocessors—A Primer .... Cohen

• Public Access

16 Computing at a Public Library .... Shair

18 Computer Power to the People .... Ahl

24 A Dream For Irving Snerd .... Nelson

27 Time For a National Computer Club .... Kuzmack

30 The Microcomputer Inflicts "Future Shock" on Technical Education .... Vuillequiz

• Computers In Education

32 Interactive Computing In Secondary Schools In France .... McLean

34 A Microcomputer Software Course .... Williams

36 Computer Science at Carnegie-Mellon University .... Hastings

38 Final Exams .... Eisenberg

42 Computational Unsolvability .... Steen

44 State-of-the-Art vs Compatibility .... Ahl

• Languages and Programming Theory

45 Something Is Missing .... Finseth

51 File Structures .... Lees

53 PILOT .... Yob

58 A Taste of APL .... Finseth

62 ARTSPEAK .... Friedman

• Computers in Medicine and Science

66 A New Generation of Biomedical Instruments... Brus

67 The Miraculous Medical Microprocessor....Weintraub

69 Computerized Robots .... Armstrong

70 Computer Correction of Optical Illusions .... Smith

73 Brown Scientists Peer Into Fourth Dimension. Norris

74 An Inexpensive Reading Machine for the Blind... Brus

75 Medical Computerized Data Bases .... Hastings

75 The Placebo and the Computer .... Hastings

• Computers in Music

76 Music Dream Machines ....Hofstetter

81 Interactive Woman Machine Improvisations

82 New Horizons for Microcomputer Music .... Wright

83 Bottom-Up Bizet .... Taylor

88 The Digital Computer: Orchestra or Composer's Assistant .... Layzer

89 The Transposition and Composition of Music by Computer .... Shmoys

• All by Ahl

90 Saga of a System .... Ahl & North

98 The First West Coast Computer Faire .... Ahl

103 Gamboling In Atlantic City .... Ahl

110 Some Tips on Using a TV Set For Computer Output ....Ahl

Fiction and Foolishness

112 The Land of Halco .... Rowlett
113 Them Hobbyists .... Dunion
114 Computer Control .... Vitale
117 Yellow Computer .... Ragen
118 Edu-Man Meets Pseudo Hero .... Ahl
119 Edu-Man Meets the Rumor Mongers .... Ahl
120 The Lighter Side of Robots
121 The Lighter Side of Computer Dating
122 Nords .... Sunstone Graphics
126 Glorobots .... Maxson
128 The Floating Point Solution .... Taylor
130 Marsport .... Sonntag
134 Out of the Mouths of Babes .... Wirth
136 Still a Few Bugs in the System

Resources

138 Compleat Computer Catalogue
142 Compendium
148 Tragedy of Errors .... Hastings
149 Input/Output
152 The Computer Tree
154 World Model Bibliography
155 Computer Music Bibliography .... Snell
158 Electronic and Computer Music
169 S-100 Computer Kit Reference List .... Purser

• Quizzes and Questionnaires

160 Computer History Trivia Quiz .... Pasahow
162 Counter Questionnaire .... Yarmolinsky
162 The Future of Computing Questionnaire .... Johnson
165 The Case of the Reader Uncovered .... Green

Puzzles, Problems, & Programs

• Puzzles & Problems

176 Puzzles, Puzzles, Puzzles
184 Maze
185 How Late Can You Sleep in the Morning? .... Ahl

• For the Calculator

186 A Picture in 20 Lines .... Young
188 Birthday Plots

• Problem Solving and Programming Techniques

189 Complex Problem Solving Experience .... Szabo & Rhoades
193 Learning by Doing .... Gruenberger
196 Algorithmic Basic .... Allen
199 Shuffling .... Jaworski
200 A Crooked Shuffle .... Filipski
203 Grid Addressing .... Akkerhuis
205 Thinking Strategies with the Computer: Piele & Wood
  Inference
  Working Backward
  Subgoals
  Trial and Error
  Contradiction

• CAI

225 Mathematics Drill and Practice .... Ahl
229 Structuring the Lesson to the Student .... Ahl
232 Further Considerations for Presenting Multiple-Problem Types .... McLaughlin

• Short Programs

236 Keeping the Loan Arranger Honest .... Warden
238 Changing Bases .... West
239 Delving Into Depreciation
239 Systematic Savings

- **Applications for Art, Music and Science**

240 Anamorphic Art........Zucker
244 Lissajous........Ruane
245 Art & Mathematical Structures ....Chandhok & Critchfield
249 Musical Magic Squares........Hofstetter
250 Scales........Thostenson
251 Eliza........North
254 A for Effort, Zero for Arab........Wiswesser
255 Inorganic Chemistry Program........Peer

Games

258 Othello........Wright
261 SWARMS........Miller
267 EUCHRE........Raybaud
271 Daytona 500........Churchill

Reviews

- **Of Calculators**

274 Sophisticated Electronic Pocket Calculators.....Tufte

- **Of Games**

276 Smart Electronic Games........Ahl
278 Comp IV........Gray

- **Of Equipment**

280 Selecting a Micro........Gray
282 Heath: Two Computers and Two Peripherals.....Gray
285 Building the SWTPC 6800........Loofbourrow
284 How I Built on Imsal 8080........North
288 Teletype Model 43 Terminal........Ahl
287 The Sol-20........North
290 Radio Shack's $600 Home Computer........Thomas

- **Of Software**

292 Software Technology Music System........Ahl
297 A Comparison of Software Systems........Ahl
297 Review of Five Small Interpreters........North
300 Notes on Languages........Chase
303 A Dynamic Debugging System........North
306 An Evaluation of Three 8080 8K Basics........North
308 New Benchmark Program........Chase
310 Two Space Games with Graphics........North

- **Of Books**

312 A Bushel of APL........Lees
320 Programming Problem Books........Sokolowski
321 Reviews........Lees & Kugel

Reflections

322 At the End of Our Third Year........Ahl

Articles and Commentary

Trends Into the Future

Two trends dominate hobby computers today. One is for computer freaks, and involves advanced hardware. Such as an Altair-compatible board that will store digitized versions of your voice in "training" mode, and then, in speech mode, when it recognizes your voice speaking one of the previously recorded words, will cause that word to be printed (this is coming up in 1977). There are already computer boards that synthesize speech. So it won't be long before computer freaks will be trying to get one computer to talk to another, not through wire, but by voice!

Other computer-freak areas involve advanced graphics, computer music, interfacing to a breadboard, digitizing the output of a TV camera, etc. So much time is spent on getting these devices to work, that very little time is actually spent by these hobbyists on computing. The emphasis here is on gadgeteering, on a constant search for the far-out and complex.

The other trend is more and more toward the average consumer's use of hobby computers. This means a certain amount of using all-on-one-board machines such as the KIM-1, EBKA 6502 Familiarizor, and EPA 68, programmed in assembly language. There are more of these all-on-one-board type of hobby computer than any other, one reason being that it's the simplest complete computer in a single package, with a minimum of parts, and is thus much easier for a manufacturer to design and produce than the more complex multi-board machines such as the Imsai 8080 or Digital Group system. For the manufacturer, there's very little labor involved, no sheet-metal work, no point-to-point wiring, and no construction manual to have to supply. A KIM-1 offers the hobbyist the cheapest way to get his feet wet, to learn the basics of computing at minimum cost, without the need for an external keyboard, or connection to a TV set or printer.

Some of these all-on-one-board computers are so simple and cheap that they'd be hard to expand, and are fine for the person who's quite sure all he wants is to learn the elements of computing without having to put too much money into a machine he might not use much after he figures out how it works.

For those who think they may want to expand their computer so as to be able to write longer programs, or to hook on an alphanumeric keyboard or cassette memory system, etc., several of these "compacts" have add-on boards. KIM-1 owners can buy the KIM-2 4K RAM memory board, or KIM-3 8K memory. KIM-4 is a 6-slot motherboard with all connectors and a regulator. And further KIMs are in the works. The EBKA expander board, which will "expand any 6502 or 6800-based microcomputer," can be bought as an empty board, or with any or all of seven options, including kits for a PROM programmer, 4K RAM, 2K PROM, baud-rate clock and interfaces for serial, parallel and dual-cassette operation.

A much more important average-consumer trend is to the wired-only computer that can be programmed in BASIC. As the hobby market appeals to more and more non-technical people, it will have to provide this high-level language, since such people will be interested in programming, and not at all in assembly language, which is too tedious and time-consuming for all but the computer freak. As it turns out, incidentally, there are *very* few hobbyists who are really into heavy assembly-language programming; most of them use BASIC.

Aimed directly at the mass computer-hobby market is the $495 PET 2001 table-top computer, with 9-inch TV screen, built-in audio-cassette unit, full keyboard and numeric keypad, 4K RAM user memory, and BASIC interpreter in 12K ROM, shown in prototype at the January 1977 Consumer Electronics Show in Chicago, and made by a calculator manufacturer (Commodore) that recently bought an IC manufacturing company (MOS Technology, makers of KIM-1). Another calculator manufacturer is said to be working on a similar home computer, although more expensive: with 32K, $2,000.

This is where the major hobby-computer market of the future lies, not in the far-out hardware, but in an all-in-one-box computer that sells for less than $1000. The user won't care if the MPU is a Zilog Z-80 or an Intel 4004. He wants to program, and he needs to be supplied with plenty of software and with plenty of tutorial material to teach him how to use the software and to write his own programs. A couple of the larger hobby-computer manufacturers are already considering hard-wired BASIC computers. This means a BASIC interpreter in some form of read-only memory. 1977 should see *several* new BASIC machines, assembled only, ready to run, for less than $500. ■

Stephen B. Gray

EFTS: Living is Better Electronically Or IS IT?

by Deanna J. Dragunas

EFTS is a magic word lately. Pronounce its single nonsense syllable in front of the appropriate congressman (Koch, Tunny, Goldwater or a host of others will do) and watch them go apoplectic on you. Pronounce it in the banking environment and see the faraway gleam in the eye of the banker which bespeaks potential profits, cost reductions, and digging out from under the paper mountain that looms daily on the horizon.

How can a simple technological concept that is in operation to some extent in a variety of scattered applications cause such emotion-laden responses?

Let's look at the simple concept and see just what it does for and to both the consumer and the purveyor of EFTS services.

EFTS, for those of you who have been out of the civilized world for three years or who have been ignoring *Creative Computing* except for the nonsense and games sections, stands for *E*lectronic *F*unds *T*ransfer *S*ystem. It is a concept that is a natural next step in the history of exchange and banking.

Back in the good old days of barter, articles of value changed hands. Granted, it was difficult to pay for something with one and a half sheep, but as long as you weren't fussy about nickels and dimes which didn't exist yet anyway, survival on this basis was a pretty simple proposition.

About the time that taxes started to make sense, at least to those collecting and living from them, the advantages of precious metals over actual transfer of goods were realized. Transportability was a big plus.

Let's skip all the intermediate steps along the way and just accept the fact that from this beginning, we made it to the point where the medium of exchange consists of "money," where the vehicles for trade are merely tokens and not in themselves worth the avowed face value. These are paper dollars and pennies and nickels and dimes and so forth.

Sending such tangible items can be both unwieldy and risky, so since the Middle Ages and the beginnings of modern banking, banks have often dispensed their clients' funds on written orders to pay. Thus, the queen of Spain could direct the payment of funds to her agents in England, Italy and France without having the gold physically toted around the Continent or across the Channel.

This practice evolved into our modern checking system. For many years, checks were physically sorted and shipped between banks. The convenience of checking accounts led to their acceptance and increased use. This made the bankers collectively smile, until they saw where it would lead them.

A projection of growth in the use of checks made for the banking industry in the decade after World War II showed that if banks continued in their merry manual way, by the 1970's every man, woman and child in the United States would have to be employed as check sorters to handle the volume.

This led to the development of those funny, semi-decipherable numbers on the bottom of your checks. They are referred to in the common jargon as MICR characters, Magnetic Ink Character Recognition characters. Instead of sorting the checks by human hand and eye, the banks now feed the checks into machines that can read the magnetic ink characters and do the sorting themselves. Never having planned a career as a check sorter, I am sure the majority of readers are as thankful for this development as I am.

Implementing the MICR systems allowed banks to do not only sorting, but also actual account manipulations without human involvement. Once the checks reach the banks who own the accounts, the amount can be keyed onto the check and the numbers are read by computers which perform the deduction or addition to the account whose number is encoded at the bottom of the check.

A few people have been more than normally grateful for this boon from modern technology. These are the individuals who quickly realized that people often come into the bank without their own deposit slips and use the desk deposit slips provided at the bank instead. Naturally, these do not have the account number at the bottom, but are blank. The account number is entered in handwriting by the depositor and back in the check-processing fiefdom, a clerk would enter not only the amount as was normally done, but also the account number.

A few enterprising individuals who had access to MICR encoding equipment removed blank deposit forms, encoded their own account numbers on the bottom, and replaced them for the unwary to use to make deposits at the bank. Naturally, when the checks were run through the computer, the account encoded at the bottom was appropriately increased.

At the end of a day or two in a busy city like New York, one's personal account could experience a massive swelling from all the deposits made by all the customers who went to the bank. The judicious thieves made a withdrawal and disappeared long before the customers complained about the inaccuracy of their next statements or before their checks began bouncing. The greedy thieves got caught.

Such problems have been solved, for the most part. But they may seem only minor interludes compared to the potential Pandora's box that EFTS presents to the eyes of some.

The use of checks is continuing to increase. This seems inevitable as long as the population or the amount of money or the amount of deferred payments in the economy continues to grow. Despite the electronic help the banks have available in sorting and debiting and crediting and producing statements, a mountain of checks must still be physically handled every day.

The next logical step to streamlining our fiscal existence is to eliminate as much paper as possible. This is what EFTS is all about.

Imagine receiving bills in the mail, going to your telephone, dialing the bank's computer and directing it by punching out the correct codes and amounts on your touch-tone telephone to move money from your account to the accounts of your creditors. Or imagine receiving a monthly statement that tells you the amount of your payroll deposits, what your utility and mortgage and phone bills were, and what your remaining account balance is. Or imagine going to a store and presenting your "credit" card and never getting a bill because entering the card into the terminal/cash register has automatically moved the payment from your bank account to the store's account. Or, more simply, imagine using a teller terminal in an airport or a shopping center or a grocery store to make deposits or withdrawals from your bank account.

Imagine any or all of these and you are thinking about Electronic Funds Transfer Systems. And all of these applications are at least on the drawing board, some of them in actual test operation.

The technology that can support EFTS systems is a combination of computers, terminals and teleprocessing. Today's large-scale computer systems are capable of storing on-line information on large numbers of accounts. Terminals are being designed in more user and applications-oriented ways: we don't fit the job to the terminal anymore, but we design the terminal for the job. And, because of our capability to send data quickly and accurately over the telephone system's communication lines and because of the more complex teleprocessing software that exists today, we can almost instantaneously update files and accounts.

Voila! Away goes the paper check problem and here comes the age of cashless convenience. Or does it?

Despite the apparent convenience of these systems, there are drawbacks, both technical and social as well as fiscal. Banks use EFTS systems today to transfer millions of dollars between themselves. An incident occurred a number of months ago that highlighted the vulnerability of such transfers to tampering. A large amount of money was transferred between banks, with all the appropriate check sums and verification data to ensure that interference in the telephone lines would not cause errors. Apparently the check sums and tallies were not as complex to fathom as the banks had hoped, and some person or persons changed the data to credit other than the directed accounts at the receiving bank. The paper confirmations of the electronic transfers lagged by the usual few days, long enough so that fund withdrawal had already been made by the thief or thieves.

A simpler case involved a direction sent ostensibly from one bank to another to pay a designated agent a specific sum of money. All was as it should be, except that the messages from that bank usually came at a different time of day. The receiving clerk noticed this, checked it out, and discovered that the bank had never sent the payment order. A knowledgeable outsider had dialed the computer. A trap was laid and the culprits were caught.

There are other little problems with current implementations of EFTS systems. Dave Ahl's item in the March-April 1976 issue of *Creative Computing* exemplifies the problems of dealing with the unusual, in his case, a foreign check, in a system designed to deal with the normal.

If we assume that the inter-bank and customer-to-bank transmission integrity problem will somehow be solved, or at least that the Federal Deposit Insurance Corporation will brace itself for a rash of insurance payments to banks and customers that have been swindled, there are still philosophical and operational problems to solve before we decide to do away with both hard cash and checks forever.

The reason that Congress and consumer and civil liberties groups are wary to the point of legal action to prevent the immediate implementation of full blown electronic funds transfer systems is the potential danger to personal privacy. An example may point up the concern better than a detailed discussion of what privacy entails.

A user of a direct funds transfer system in which a credit card type of authorization is required for each transfer would probably receive a periodic, perhaps monthly, statement of account giving at least the remaining balance. In case of disagreement between the individual and the bank, some sort of record must be available to back up the bank's claim against the customer's claim of what the appropriate balance should be. To be useful, this would have to be a detailed listing of times and places of financial transfer authorizations. Had a well-meaning gentleman requested a statement, he and his wife might discover that there is an entry that shows the husband checked into a San Francisco hotel with his wife on the same day that his wife purchased groceries in Boston. Obviously, an error has been made. Whether it has been a human error in judgment by the husband, or an electronic error or potential computerized theft which is the bank's responsibility remains to be determined!

However, the fact is that to ensure accuracy and auditability, the bank must collect what amounts to a dossier on the activities of its customers that rivals that even thought about by the CIA or FBI and kept in only sketchy form today by credit card companies.

What had been a technically feasible, operationally desirable approach to speeding cash flow has become a question for the courts to settle. How much information must be kept to ensure accuracy and auditability? What information may not be kept? How long may or must information be kept? Very importantly, who can see the record of cash transfer activities besides the account-holder?

Dave Ahl's article, "The Magic of Electronic Funds Transfer" in the March-April 1976 issue also brings up another problem, a problem of monopoly. All banks may not have EFTS systems. And all banks do not offer the same services. Foreign checks could not be cashed by the banks which handled the *Creative Computing* account because of a change in foreign exchange regulations. But because of the processing that was done by the interbank EFTS system, *CC* was charged for *not* processing these checks. The EFTS charge was relatively large, almost as large as the amount of the checks.

Initial EFTS systems will be very expensive to install and support. However, they will be attractive for both their novelty and their convenience. Only large chains of retailers will be able to absorb the costs and still retain competitive prices on merchandise. This means carrying money or using credit cards at smaller establishments or simply avoiding these places and giving the larger businesses a competitive advantage.

Another problem may present itself to the consumer of today's credit. There are some people and some businesses that remain solvent only by the grace of the delay between the time a check is written and the time their accounts are finally debited. This is called a "float." Having worked for at least one now successful business which would have gone down the tubes without the grace of paper delay, I expect some resistance to certain potential implementations of EFTS systems.

For instance, can a bank grant a mortgage and require that an authorization for a direct EFTS payment of the monthly mortgage be a condition of the loan? It is not the payment itself which is in question, but whether such a requirement infringes on individual liberties, in this case the right of the individual to live as dangerously close to insolvency as possible.

EFTS is now at that formative controversial point where consumers, technologists, Congress, and the courts are trying to decide not only what types of technology may implement EFTS, but also what type of world it will help us create. ■

The World In Your Own Notebook

John Lees

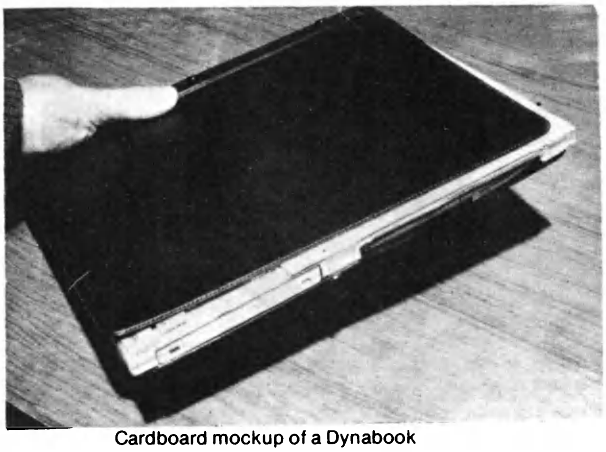
Imagine having your own self-contained knowledge manipulator in a portable package the size and shape of an ordinary notebook. Suppose it had enough power to outrace your senses of sight and hearing, enough capacity to store for later retrieval thousands of page-equivalents of reference material, poems, letters, recipes, records, drawings, animations, musical scores, waveforms, dynamic simulations, and anything else you would like to remember and change.

Such flights of imagination are what one would expect to find as the basis for a well-written science- fiction novel, or perhaps as the musings of a Creative Computerist wishing she could carry the school computer system to the park and use the text-editing facilities to write poetry while sitting under a tree. It isn't often that such an idea is the basis for a serious research effort by a major company, in this case the Learning Research Group of the Xerox Palo Alto Research Center (PARC), which recently released its latest report on Personal Dynamic Media—the Dynabook.

Actually, Alan Kay's original draft note (August, 1972), suggesting that PARC conduct research into the effects of personal dynamic media, did begin with the rejoinder that it should be read as science fiction. This was appropriate caution, since the newly marketed Hewlett Packard HP-35 was just then giving a first glimpse into how quickly miniaturization would put large amounts of data-handling ability into very tiny packages, and it was still two years before MITS announced the Altair. So one might very well think of the research on Dynabook as being applied science fiction; the investigation of a plausible "what if."

The Learning Research Group at PARC sees in the power of computing the ability to provide a new kind of media. Until now, all media have been essentially passive. Newspapers, books, films, radio, television: all are media forms which one may watch, but not interact with, not participate in on an individual basis. The few specialized exceptions to this, coloring books, primitive computer time-sharing systems, etc., have been limited at best. What was desired was a flexible, dynamic, active, personal medium which could help a person to learn about, interpret, and interact with the world.

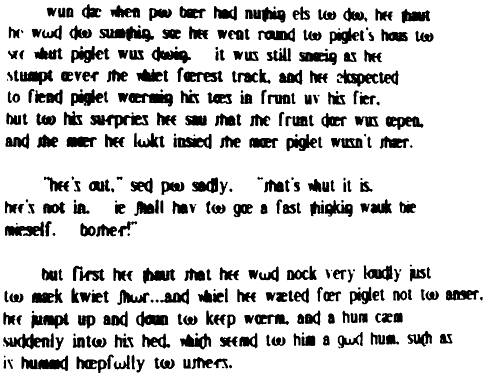
So a new technological device was designed; a device not yet possible to manufacture but within the reach of present technology: "The size should be no larger than a notebook; weight less than 4 lbs.; the visual display should be able to present at least 4000 *print-*



Cardboard mockup of a Dynabook

An interim Dynabook

Some of the fonts possible

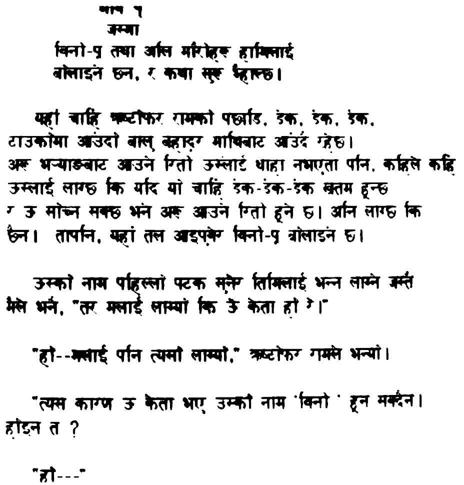


wun dæ when pω bær had nuthig els tω dω, hee thaut
he wωd dω sumthig. sœ hee went round tω piglet's hous tω
see whut piglet wus dωig. it wus still snœig as hee
stumpt œver the whiet ferest track, and hee ekspected
to fiend piglet werming his tœs in frunt uv his fier.
but tω his surpries hee sau that the frunt dœr wus œpen,
and the mær hee lωkt insied the mær piglet wusn't thær.

"hee's out," sed pω sadly. "that's whut it is.
hee's not in. ie shall hav tω gœ a fast thigkig wauk bie
mieself. bother!"

but first hee thaut that hee wωd nock very loudly just
tω mæk kwiet shωr...and whiel hee wæted fer piglet not tω anser,
hee jumpt up and doun tω keep werm, and a hum cæm
suddenly intω his hed, which seemd tω him a gωd hum, such as
is hummd hœpfωlly tω uthers.

ITA



भाग १
उम्मा
विनो-पू तथा अलि मरिहेक हामिलाई
बोलाइने छन, र कथा सुरु हुन्छ।

यहाँ चाहिँ क्रिष्टोफर रबिनको पछाडि, डक, डक, डक,
टाउकोमा आउँदो बाल् बहादुर माथिबाट आउँदै गर्छ।
अरु भन्याङबाट आउने रितो उस्लाई थाहा नभएता पनि, कहिले कहिँ
उस्लाई लाग्छ कि यदि यो चाहिँ डक-डक-डक खतम हुन्छ
र उ सोच्न सक्छ भने अरु आउने रितो हुने छ। अनि लाग्छ कि
छैन। तापनि, यहाँ तल आइपुगेर विनो-पू बोलाइने छ।

उस्को नाम पहिल्लो पटक सुनेर तिमिलाई भन्न लाग्ने जस्तै
मैले भनें, "तर मलाई लाग्यो कि उ केटा हो ?।"

"हो--मलाई पनि त्यसो लाग्यो," क्रिष्टोफर रबिनले भन्यो।

"त्यस कारण उ केटा भए उस्को नाम 'विनो' हुन सक्दैन।
होइन त ?

"हो---"

Sanskrit

ing quality characters with contrast ratios approaching that of a book; dynamic graphics of reasonable quality should be possible; there should be removable local file storage of at least one million characters (about 500 ordinary book pages) traded off against several hours of audio (voice/music) files." It is envisioned that a Dynabook will cost about $500, which should be inexpensive enough for school systems to supply Dynabooks free to their students, since textbooks would be replaced by Dynabook removable files.

Since this ideal is not yet technologically possible, the Dynabook is being implemented with existent hardware, refered to as interim Dynabooks, so that the effect of the Dynabook concept on learning and education can be studied. The interim Dynabooks meet essentially all of the hardware objectives except size and cost and they provide the opportunity to develop the all-important software for Dynabook. Given the present state of the art, software development is much more difficult and time-consuming than hardware development. Dynabook will eventually be put together from more or less "off the shelf" hardware components, but the software which will give life to the concept must go through a long and arduous process of development if it is to aid and not hinder the goals of a personal dynamic medium.

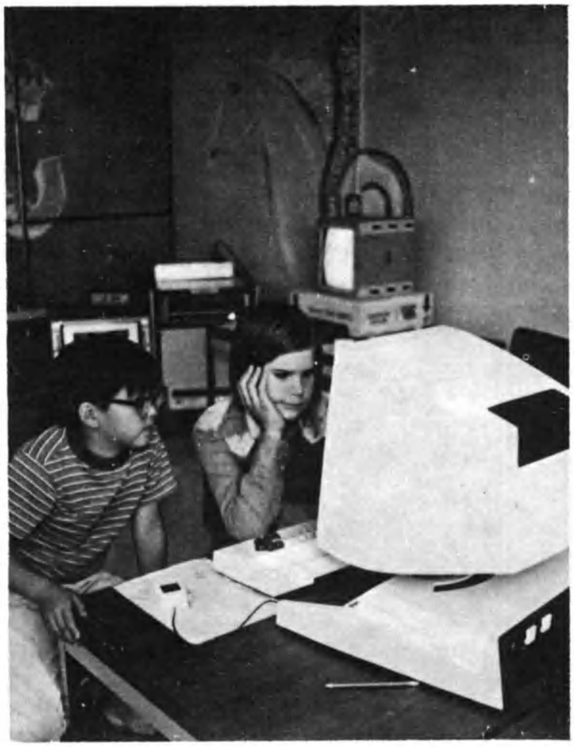
The software system being developed for Dynabook is known as Smalltalk and its capabilities are truly amazing. [See "Learning About Smalltalk," Sep-Oct 1975 and "A Smalltalk Airplane Simulation," Mar-Apr 1976 *Creative Computing*.] With the graphic capabilities of an 8½ by 11″ display composed of over one million points, some really fantastic things are possible, from the planned book-quality printing in any desired font up to animated cartoons, musical scores that can control a synthesizer, all manner of simulations, the ability to "paint" on the display as if it were a canvas, and just about anything else an imaginative person can think of!

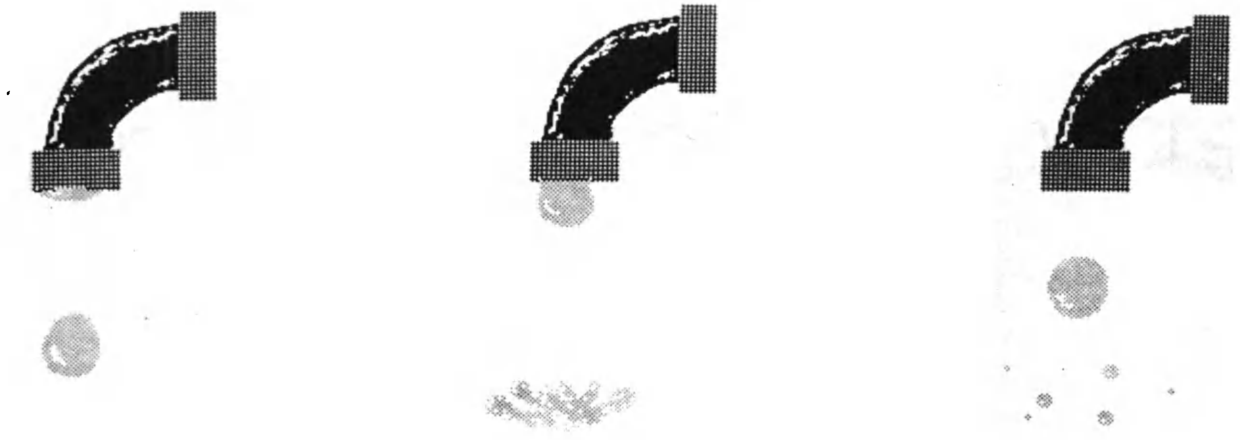
Smalltalk is a very powerful yet easy-to-use language which deals with objects in process. The idea is that "simple things should be *very* simple (while not constraining later expert use) and complex things should be *very* possible." Rest assured that Smalltalk is like nothing you've every seen before! Implemented at present only on the interim Dynabooks, Smalltalk itself is going to cause an upheaval when it gets out into the world.

So there you have it; a short introduction to an idea which may change your world. Dynabook could very easily be a reality by the end of this century. It isn't too far-fetched to say that the much touted "computer age" will truly have arrived when something such as the Dynabook becomes common. At the present time computers have not so much changed our lives as they have made it possible for existing institutions to continue to exist in a rapidly expanding world. Banks, for instance, would have become impossible ten years ago without the aid of computers, but only now with the first uses of Electronic Funds Transfer systems is the basic character of the institution of banking beginning to change.

The Dynabook concept has the potential to affect in a very basic way a great number of our society's institutions. The impact of the next new medium, Dynabook, could be earthshaking.

Look at what will be affected by widespread use of Dynabooks and particularly by widespread use of Dynabooks in networks, via phone lines or cable television systems. For *correspondence* (and the postal system)—the Dynabook has all the capabilities of a typewriter in a smaller, more versatile package, and can send and receive letters in addition. In *publishing,* Dynabook is a personal printing press. Your textbooks can be a memory file, your family can receive its newspapers on a reusable file, its monthly magazines on another reusable file. Forests will have a chance to





Dripping Faucet

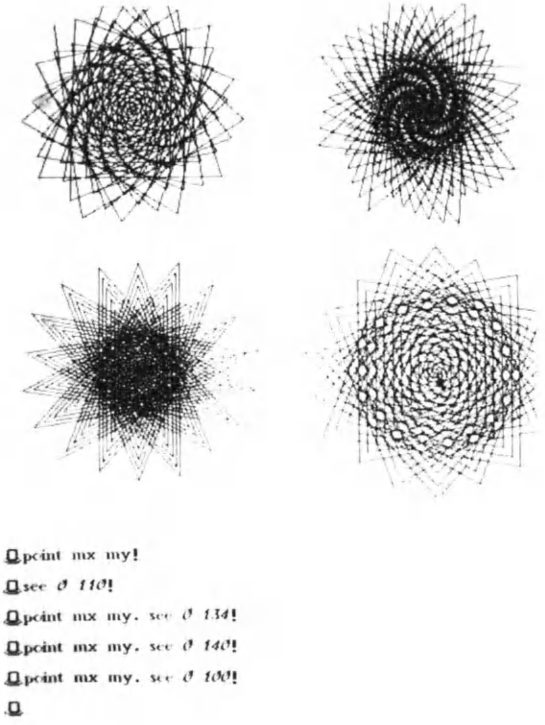
grow again! Someday you will be able to access the Library of Congress by plugging into a network. In *education*, the new learning activities possible are boundless, and they will depend less and less on dedicated school facilities. As for *calculators, televisions, radios,* all would face a very stiff challenge. Television in particular would be in for trouble—who's going to watch poor television programs if they can link up with all the other kids on the block for a game of really super Startrek or Startrader? The possibilities boggle the imagination!

It's all been said before, you say. People are always saying great things will come of computers and nothing ever happens. Bah, humbug!

Oh, yeh? Don't forget we're talking about *personal* computing, not about a monster machine somewhere with terminals all over the place. That's the difference. Look what happened with hand-held calculators. The same thing could happen with Dynabook. The important thing to realize is that Dynabook would be *distributed computing* with *centralized information.* The cost of making information and knowledge available is going up fast. In that climate, distributed hardware and centralized information make worlds of sense. Programming is expensive. Let each individual do the programming!

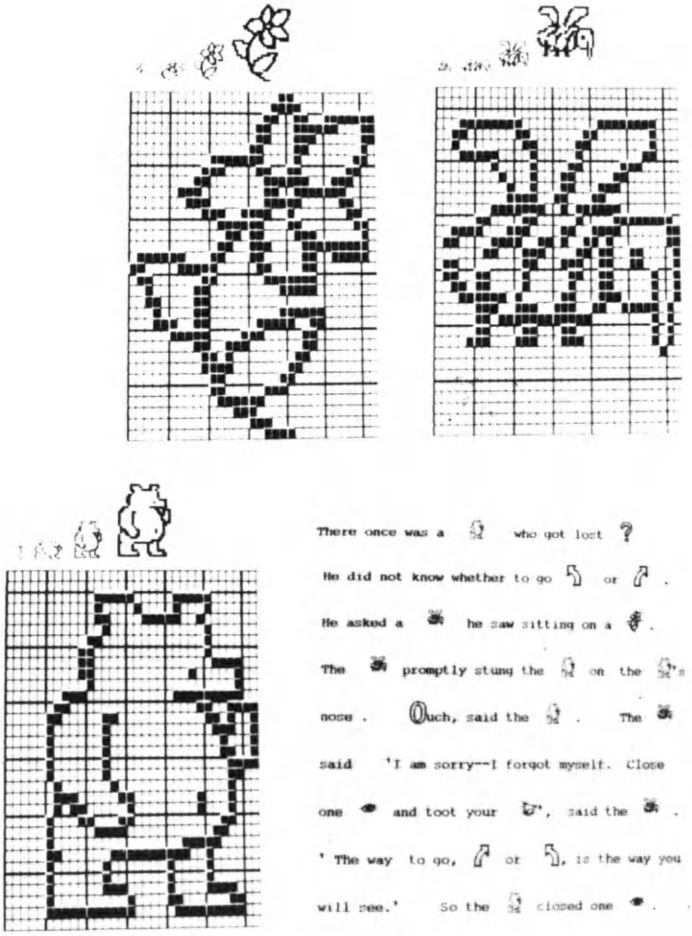
Maybe Dynabook is just a dream. I certainly hope not. Many's the time I've wished I could wander around or go sit under a tree with the book I'm reading and a sketchpad and a typewriter but haven't done so because I didn't want to pull a wagon. Maybe someday I'll be able to just pick up my Dynabook and walk out the door. It sure would be nice. I'll keep on hoping; some dreams do come true. ■

Thanks to PARC for permission to use quotes and photographs.



```
point mx my!
see ϑ 110!
point mx my. see ϑ 134!
point mx my. see ϑ 140!
point mx my. see ϑ 100!
```

Example of Smalltalk



There once was a who got lost ?
He did not know whether to go or .
He asked a he saw sitting on a .
The promptly stung the on the 's nose . Ouch, said the . The said 'I am sorry--I forgot myself. Close one and toot your ', said the .
' The way to go, or , is the way you will see.' So the closed one .

Telling a story with pictures

Eeny, Meeny, Micrc And More

Alan B. Salisbury

Until the relatively recent arrival of the microprocessor and microcomputer on the scene, "personal computing" has been largely limited to the privileged few with access (authorized or "bootleg") to the computer facilities of their employers or the computers in their schools, colleges and universities. A mere handful could be found who could either afford to buy their own minicomputer or were resourceful enough to construct their own equipment.

This picture is rapidly changing. As the readers of *Creative Computing* are well aware, the age of the affordable computer has already arrived for many and will soon be here for the rest—all thanks to the microprocessor. For the computer hobbyist considering buying or building a personal computer, there are many factors which should be taken into consideration.

Some Definitions

First, the distinction between a micro*processor* and a micro*computer* should be clearly understood. A microprocessor can be simply defined as a central processing unit (CPU) on a single LSI chip (or, in some cases, set of chips). As illustrated in figure 1, the CPU consists of the arithmetic and logic unit (ALU) with its working registers, and the control unit of a computer. It therefore does *not* include the main memory or the input/output driving circuitry and interfaces. Earlier microprocessors even excluded the clocking circuitry from the basic CPU chip.

To qualify as a micro*computer,* the total hardware, a micro*processor* may be available as a single LSI chip, while a micro*computer* may be available on a single card. We can carry this one step further and define a microcomputer *system* as a microcomputer plus the required supply, control panel (this may be as little as an on/off switch), chassis or cabinet, and some (at least minimal) input/output devices.

Microprocessors

With this perspective, one can now appreciate that a $19.95 microprocessor is a long way from being a working computer (typically, at least several hundred dollars away). Still, within every microcomputer there beats a microprocessor heart that gives it its "personality." The implications of this are many and some of these will be discussed later in this article. For now, let's take a closer look at the types of microprocessors commonly found.

It was mentioned earlier that the CPU may be on a single chip or made up of a set of chips. Single-chip CPU's are most common today. They accommodate a fixed word size of 4, 8, 12, or (recently) 16 bits, and have a fixed (predefined by the manufacturer) instruction set.[1] Both binary and binary-coded-decimal modes can be found, and the total number of different machine language instructions available is on the order of 100. Typical instructions execute in several microseconds.

A good example of this type of microprocessor is the popular Intel 8080 (see Figure 2). The 8080 is an 8-bit microprocessor with 78 different instructions, packaged in a single 40-pin dual in-line package (DIP). Pin-compatible 8080's are also available from other sources in addition to Intel. Newer versions of the 8080 operate faster, require less external support, and some have expanded instruction capabilities.

For comparison, Table 1 illustrates characteristics of several of today's more popular microprocessors. These are the ones

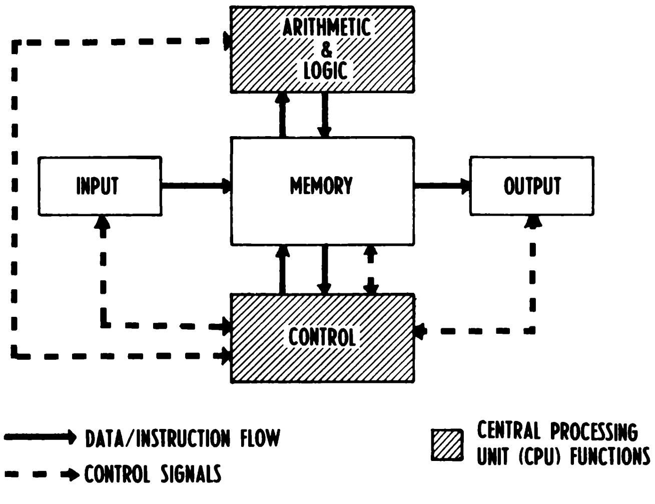


FIGURE 1: FUNCTIONAL BLOCK DIAGRAM OF A TYPICAL COMPUTER

found in many of the microcomputer systems and kits currently on the market. The Z-80, in particular, is interesting because of its relationship to the 8080. While it is not pin-compatible with the 8080 (that is, you can't unplug one and replace it with the other), all 8080 instructions are also present in the Z-80, and therefore an 8080 program will run on the Z-80. The Z-80 can be viewed as a "super" 8080 with a lot of expanded capabilities. (Note, however, that a Z-80 program will *not* necessarily run on an 8080!)

---

[1] Suggested background reading for those not familiar with these terms is "Beyond Basic" in the Nov-Dec 1976 issue.

In addition to the basic CPU chip, each micro is complemented by a set of available memory and I/O interface chips. Caution must be taken in selecting add-on memories and such to insure that chips are compatible with one another (that is, common logic voltage levels and the like), or, in the case of complete cards, that they will properly interface with one another. True industry-wide standards do *not* yet exist in this area.

"Bit-Slice" Micros

A separate class of microprocessor chips is referred to as "bit-slice." Each bit-slice chip contains an elemental portion (for example, 2 bits) of an arithmetic and logic unit plus a similar of the working registers. An 18-bit CPU (Figure 3) could be assembled in this case by interconnecting 9 register/ALU chips and adding a separate chip for the control unit. Machines of arbitrary size can be built in this manner.

The control units for bit-slice micros usually do not have predefined fixed instruction sets. The detailed step-by-step execution of an instruction is governed by the information contained in a separate control memory. In effect, this kind of control unit is a "computer within a computer" and the control memory contents are referred to collectively as a "microprogram."[2] It is therefore possible for the user to define his own instruction set, or to "emulate" (that is, copy) the instruction set of another computer in order to use the same software.

Examples of bit-slice chip sets are the Intel 3000 and AM 2900 series devices. Each series includes many devices to provide the capability of building very sophisticated computers, probably more "mini" than "micro" in performance and complexity. Bit-slice micros are best left to the engineer or the hardware oriented hobbyist.

Technology

Most of today's commercially available microprocessors utilize n-channel metal-oxide-semiconductor (NMOS) technology. MOS technology is the technology associated with the "field effect" transistor. NMOS, although slightly more complex than PMOS, offers a decided speed advantage over the latter.

The fastest microprocessors available are generally of the "bi-polar" transistor-transistor logic (TTL) type. Bi-polar technology is that used in the common PNP or NPN transistors. The density of these devices (that is, the number of equivalent transistors that can be placed on a single chip) is considerably less than MOS and the power required is higher. On the other hand, they operate at much faster speeds. For these reasons, bipolar technology is often used in the bit-slice class of micros.

A relative newcomer in the field is "integrated injection logic" or I2L, a relative of bi-polar technology. I2L promises densities and power requirements comparable to MOS, with speed even better than MOS.

Memories

Two types of memories may be used within microcomputers: "read/write" and "read-only." Read only memories (ROMs), as their name implies, may be read but not written (altered) under program control. ROMs are most often employed in microcomputer systems that are dedicated to a single application such as a process controller. The program in this

FEATURE: / MICRO PROCESSOR:	Word Length	Technology	Number of Instructions	Instruction Execution Time (microseconds)	Registers	COMMENTS/NOTES
FAIRCHILD F-8	8-bit	NMOS	74	2	Accumulator **(Note 1)** 64 Gen Purpose	1. Indirect addressing used to reach all but 12 Gen purp registers.
INTEL 8080	8-bit	NMOS	78	2	Accumulator 6 Gen Purpose Stack Pointer	8085 integrates support chips with CPU, is faster. PL/M language
INTERSIL 6100	12-bit	CMOS	60+	5	Accumulator MQ	Emulates PDP-8 instruction set!
MOTOROLA 6800	8-bit	NMOS	72	2	2 Accumulators Index Stack Pointer	Instruction set is similar to PDP-11. MPL language
TEXAS INST 9900	16-bit	NMOS	69	2.7	(Note 2)	2. Uses memory-to-memory instructions, without CPU working registers.
ZILOG Z-80	8-bit	NMOS	158	1.6	2 banks	Super 8080, software compatible (upward 8080 to Z-80) but not pin compatible. PL/Z language.

TABLE 1. A Comparison of Several Popular Microprocessors

---

[2] Note that "microprogramming" is *NOT* simply the same as programming a microprocessor! Generally, a *microprogram defines* the instruction set, *while the machine language program uses* those instructions.

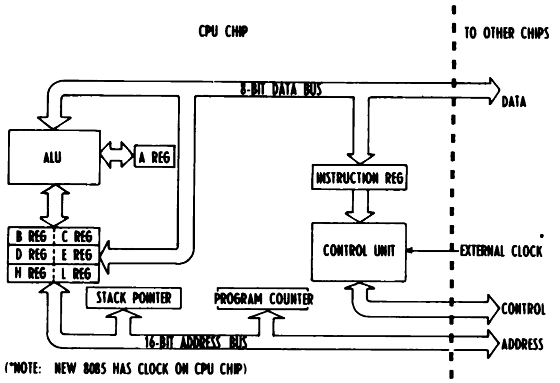


FIGURE 2· INTEL 8080 CENTRAL PROCESSING UNIT

type of application normally remains constant. A small amount of separate read/write memory is often included for data which may be variable. Another use of ROMs is for the control memory of a microprogrammed computer as described earlier. In this case they are often referred to as "firmware." Finally, key systems programs (monitors, interpreters, etc.) are sometimes provided by microcomputer manufacturers in ROM form to eliminate the necessity of having to read them into memory.

General-purpose microcomputers of interest to the computer hobbyist require read/write memories for both programs and data. Read/write memory chips are usually called RAM (in contrast to ROM), for "random access memory."

RAM's may either be "static" or "dynamic." Dynamic RAM's have the disadvantage of requiring a periodic "refresh" or they will lose their information, and this requires extra circuitry. "Volatile" memories of this type lose their contents when power is cut off, just as most pocket calculators do. Unless the system has fairly high-speed input devices for loading memory, or keep-alive batteries, volatile memories leave a lot to be desired.

A key factor concerning memories is capacity. Usually memory is available in increments of 1K (K = 1000) words. From the hardware standpoint a system should be able to accommodate additional plug-in boards to expand memory (that is, physical space in the cabinet, plus power), and the boards must electrically interface with the CPU. The more popular micros have already seen independent companies providing "plug-compatible" memories for their products. From the software standpoint, a micro may be limited in the amount of memory which its instruction set can address, but that limt is generally considerably higher than most hobbyists will require.

Connections between IC chips (CPU, memory, input/output interfaces, etc.) normally utilize "buses." A bus is simply a parallel set of lines grouped together as a set. More than one device can be connected to a bus at the same time with addresses or "select" lines used to cause the desired one to respond while others ignore a signal. Separate buses may be used for addresses, data, or control functions. In the case of memories, a CPU could, for instance, place a memory address on an address bus, a "read" command on a control bus, and receive the contents of the desired memory location on a data bus.

Input/Output and Peripherals

One of the biggest problems faced by the computer hobbyist is finding suitable input/output devices at a reasonable price. Those who have spent much time on commercial minis or larger computers find the performance of the affordable range of input/output devices somewhat disappointing.

Two general types of I/O interface are provided with most microcomputers. Serial interfaces (one bit at a time, sequentially) are probably the most practical since they can be used with more common input/output devices available for the hobbyist. Parallel interfaces (multiple bits simultaneously, usually 8 to form a complete character) are more powerful but require greater equipment sophistication.

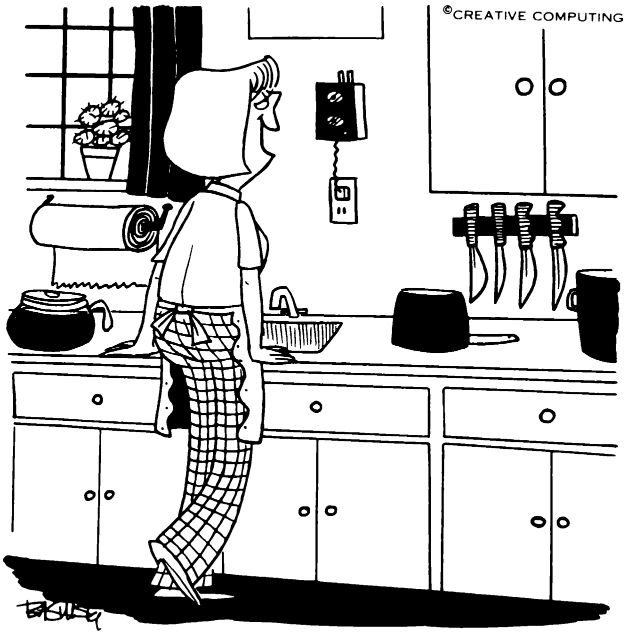
The most economical input device available is a simple alphanumeric keyboard. These are widely used and most microcomputers have suitable interfaces either as a standard feature or as an option. A keyboard is ideal for interactive work such as working with short programs in BASIC. They can be quite frustrating, however, when used for loading long programs since they are limited in speed to the user's typing speed. On the other hand, a full alphanumeric keyboard is far better than a limited numeric or hexadecimal (0 thru 9 plus A thru F keys) keyboard or set of sense switches; these devices require entry of data either in binary form or 4 binary bits at a time encoded into hexadecimal.

On the output side, video terminals are both effective and practical. Their practicality stems from the fact that it is possible to use a normal home TV set for this purpose without any modifications. Typically 1024 characters can be displayed on the screen at any time. Again, this is a very effective output medium for interactive use. The disadvantage here lies not in speed, but in permanence, since no "hard copy" is available. As with keyboards, interfaces for video output are generally available as options for microcomputers, and in some cases are standard.

Medium and high-speed input/output devices and "hard-copy" terminals are still prohibitively expensive for most hobbyists. Surplus teletypewriters (with or without paper-tape readers and punches) are one of the better buys, but even so they probably cost two to three times the price of the basic microcomputer. Also, except for the mechanically minded, they can present troublesome maintenance problems.

Cassettes

Cassette tape recorder/players perhaps provide the light-on-the-horizon of the input/output and peripheral dilemma for the hobbyist. While special digital cassette tape drives have been developed for the computer industry, normal audio-cassette recorders are proving to be very satisfactory for personal computing. Even the less expensive devices with reasonable quality audio tape work well. There are a number of techniques for handling digital data on audio tapes and, to date,



©CREATIVE COMPUTING

"Computer, computer on my wall... who's the fairest of them all?

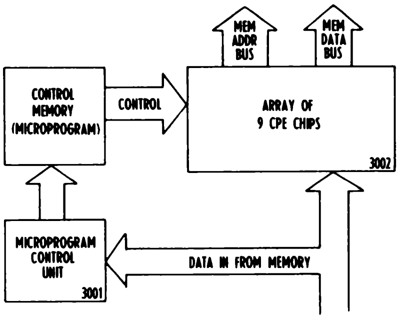


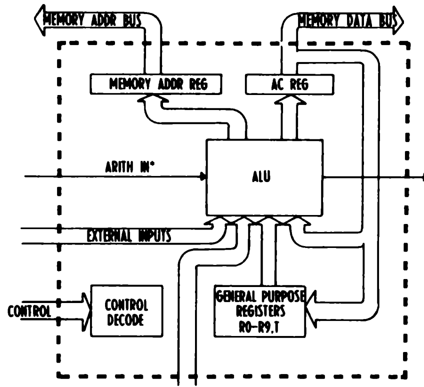
FIGURE 3b: INTEL 3000 SERIES CONFIGURED AS 18-BIT CPU

there are no real standards. Nevertheless, a few suitable techniques have been published and are finding wide acceptance for hobbyist use.

The utility of cassettes lies in their ability to provide a high-speed (compared to keyboard) input medium and permanent storage for retaining long programs after they have been developed. Standards further permit the exchange of programs in machine-readable form between those using the same standard.

Support Systems

Several types of support systems are marketed to support development of microprocessors for industrial and commercial applications. These are aimed primarily at the developer of systems in which the program will be implemented in ROM for a fixed application. The idea is to permit use of read/write memory during development so that the expense and inflexibility of ROM's can be delayed until all the bugs are out of the program.



*CARRY ETC FROM PREVIOUS/TO NEXT BIT SLICE

FIGURE 3a: INTEL 3002 CENTRAL PROCESSING ELEMENT (2-BIT SLICE)

Emulators (sometimes called in-circuit emulators) are essentially complete microprocessor or microcomputer systems with RAM instead of ROM. When a microcomputer with ROM is to be embedded in a larger system, an emulator can be inserted in its place with its program in RAM to permit checkout of the overall system.

Another way to test programs is by using a simulator. A simulator is usually a program that runs on a larger computer (often a time-sharing system) that simulates detailed execution of a microprocessor program. It accepts microprocessor machine-language or assembly-language programs as input and produces the same output which the actual microprocessor would, sometimes with diagnostic information included to help find program bugs.

Software

Probably the single most important factor that should be considered in selecting a microcomputer for personal use is software. With very few exceptions, software written for one microprocessor type will not operate correctly on a different type. For example, an Intel 8080 program will not run on a Motorola 6800. Therefore, careful thought should be given to the microprocessor which will be the CPU of your microcomputer. Its instruction set and registers make up its unique "personality" alluded to earlier.

Staying with almost any of the mainstream microprocessors will ensure availability of a fairly wide range of software, some from the manufacturer of the chip itself, some from the microcomputer manufacturer, and some from other users and independent developers. There *are* differences in what's available, however, and it would be well to consider these according to individual needs and desires.

Systems software availability will to a large extent determine the limitations of a microcomputer system. The various kinds of systems software were fully described in the Nov-Dec 76 issue of *Creative Computing,* and most of the program types described in that article are applicable to microcomputers.

Many versions of BASIC have been developed specifically for microcomputers, and they have varied capabilities according to the amount of memory available in a system. Minimal memory sizes of 1-2K words are required for almost any systems software, with 4K really providing a baseline capability. Once again, readers are cautioned to beware of incompatibilities between BASICs, even though on the surface they may appear to be the same. BASIC for micros is usually of the interpreted variety rather than compiled.

Compilers and assemblers with much sophistication are generally "cross" compilers and "cross" assemblers; that is, they compile or assemble machine-language programs for a particular microprocessor, but they themselves run on a larger computer, perhaps accessible through a time sharing system.

Few true compilers for microcomputers are around today. The predominant language used for those that are available is PL/M, a derivative of PL/1 pioneered by Intel. A similar (not identical) derivative also beginning to appear is MPL from Motorola.

Operating systems for low-end personal microcomputers are rather primitive unless considerable memory is available. Monitor facilities which aid in checking out programs are frequently found, often implemented in ROM.

Summary

Moving from a terminal on to your own truly personal computer can open up a whole new world of fun—and challenge—FOR THE COMPUTER HOBBYIST..Whether you decide to build your own or buy an off-the-shelf microcomputer, you should plan ahead, well beyond the system you initially obtain. As your capabilities and desires expand, so must your hardware and software. A carefully chosen system will be able to evolve along with your needs.

The "big picture" in the microprocessor/microcomputer arena is continuously changing. Many 16-bit microprocessors are either in production or already announced. Some are even micro**computers** on a single chip, including one with the full NOVA instruction set! Another consideration is that TV games are rapidly approaching the classification of personal computing, as the newest programmable systems from Fairchild and RCA have demonstrated. For some, this may be the best way to get into personal computing.

Certainly it can be anticipated that new IC developments will soon be showing up in the assembled microcomputer and microcomputer kit marketplace. Waiting for this to happen may ensure you never get your own computer though, because in this field there will always be something significantly better "just around the corner." ■

The Pocket Computer is *(Almost)* Here

Richard A. Ahern*

In at least two semiconductor laboratories in this country, technical obstacles have already been overcome and production problems are now being solved on a battery-powered, pocket-sized general-purpose computer that can:

1) sequentially access up to 13 million bytes of data from a self-contained half-ounce high-data-density microcassette,

2) randomly access (or nearly so) up to 100,000 bytes of data from an internal continuous-loop tape,

3) show up to 378 characters simultaneously on a low-power 2½ × 2½-inch thin-film "flat screen" display,

4) display graphics and slow-frame-advance low-resolution video on the same 128 × 128-dot dot-matrix screen, and

5) send and receive at up to 1200 bauds.

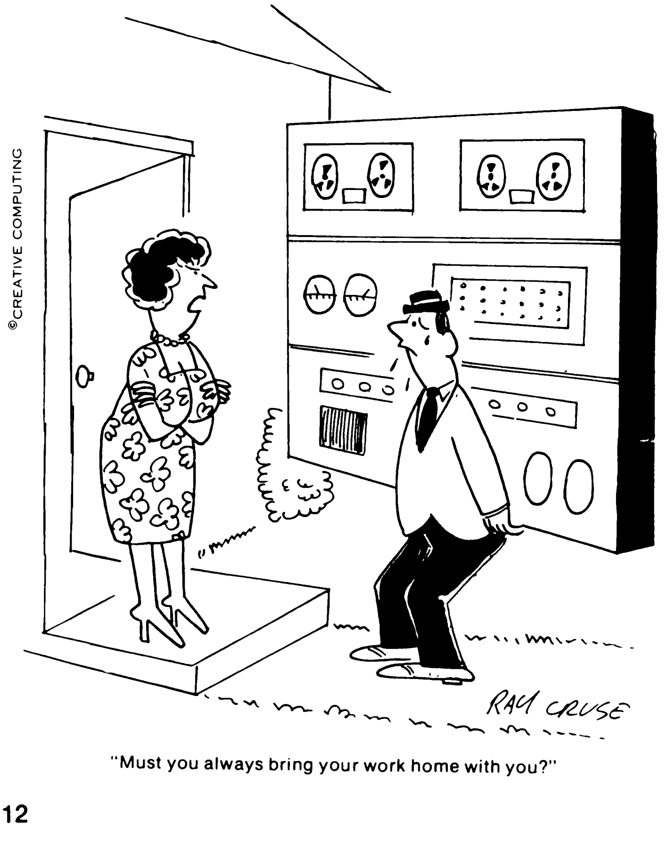
Introduction

The pocket computer will be able to do much of what many general-purpose desk-top computers now do, but will be almost as small and as inexpensive as the more advanced of the recently introduced programmable pocket calculators.

The device itself, in its most all-inclusive configuration, will look like a pocket dictating machine with a calculator keyboard and a miniature TV screen on its face. Although it will have only several more keys than many programmable calculators, it will be capable (with a shift key) of full USASCII character generation. Although its screen will resemble a small TV screen (2½ × 2½-inches, or so), it will actually be a lightweight (3 to 4-ounce), X-Y-addressable thin-film dot-matrix display (probably either 128 × 128 or 256 × 64). The long-awaited "flat screen" display was finally introduced (with production commitment) at the December, 1975, International Electron Devices Meeting.

The high-data-density microcassette subsystem, although less eagerly awaited (indeed, hardly talked of at all within the industry until its recent advent), is nevertheless the key to the power of the pocket computer. The common wisdom has long seemed to have had it that to achieve an error rate of 10^{-6} bits or better on a cassette system, there should be no more density than 800 BPI on only two tracks per cassette. However, a well-known nonprofit research organization has recently developed a production-ready cassette subsystem with 7000 BPI and 8-track-per-cassette densities that has an error rate that is still 10^{-6} bits.

Furthermore, work is now being done to develop a continuous-loop microcassette system capable of accessing up to 100,000 bytes of data at an average access time of 900 milliseconds (based on 15 inches of continous-loop tape moving at 8½ ips at the above densities). Such a system would be able to replace or supplement high-ROM-need systems and would be able to supplement semiconductor RAMs in almost any random-access system, allowing subroutines in a program to be called in (or, indeed, even entire sequential programs), one at a time, and reducing the need for semiconductor RAMs.



"Must you always bring your work home with you?"

---

*Applied Electronics, P.O. Box 161, Forest Hills, NY 11375

Conventional Uses

The pocket computer will be able to be used for many of the same administrative purposes that intelligent terminals and small general-purpose computers are now required for, especially order entry and other "fill in the blank" operator-prompt/data-check remote batch-processing jobs. It will be especially helpful when a user wishes to take it from place to place (or home at night) for it should weigh only 20 ounces or so, not 40 to 60 pounds as its counterparts of today do. In addition to its administrative functions, the pocket computer will be usable in many scientific applications; those that do not require extremely fast memory functions.

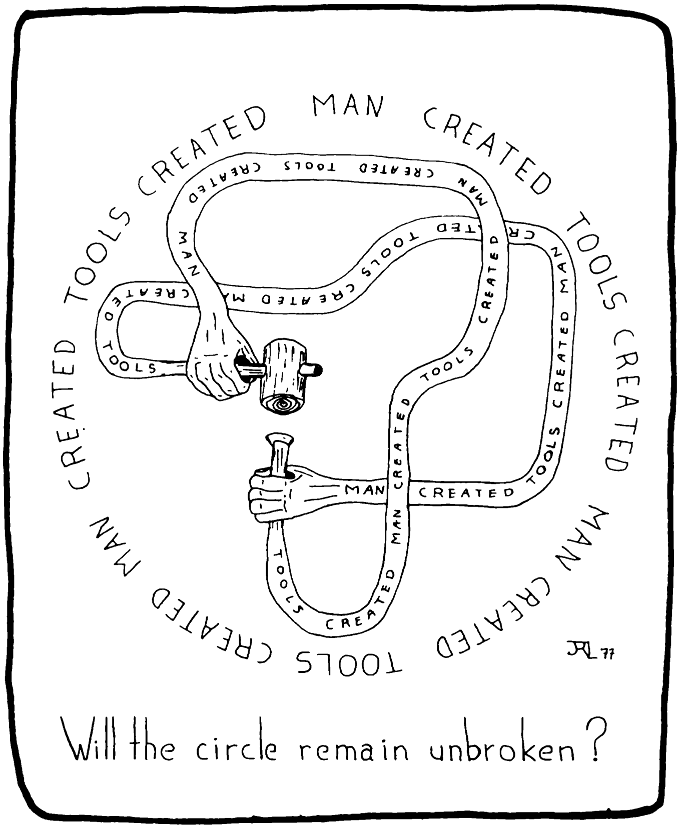
The device will probably not have a printer as part of it, but hardcopy—either selective or entire reports—will be able to be easily obtained by holding the pocket computer's "send" telecommunication port to the telephone mouthpiece after first calling any ASCII computer, printer or Teletype (speeds of 1200 bauds can easily be achieved with small microelectronic battery-powered acoustical couplers). Alternatively, the microcassette could be removed from the pocket computer and brought to any computer station with a microcassette reader. (Which we will see many of as pocket computers are introduced).

Technical Problems

Like pocket calculators, pocket computers will be designed to be able to be put together by hand in 10 to 20 minutes. Although there will be a troublesome interconnect problem—especially with display screens of 128 × 128 dots or larger—the two most significant technical problems lie in fully debugging the microcassette (especially the continuous loop) and display subsystems.

Regarding the microcassette system, reading or writing continuously at high tape densities is not a problem. Provided the address is not too long, the tape blocked by record and the tape speed kept down, searching can also easily be done. However, frequent starting and stopping followed by selective reading and writing can become more difficult (in high start/stop systems that require only data checking and indications of nonstandard conditions, analog pulses can be added to the tape). The continuous-loop microcassette, when used only as a ROM replacement or addition, should present no problems. Problems will arise when too much writing has to be done and complex housekeeping routines have to be set up. If high-density selective writing results in too high an error rate, low-density writing could be used for RAM or, in some cases a system of double addresses and an either/or search system could be used to reduce the error rate.

There are also problems with the display subsystem, but again none that are not able to be solved. The cadmium selenide electroluminescent phosphorus thin-film display which has been most discussed recently requires a voltage that is probably too high for a hand-held device. Electrochromic displays, although they add the prospect of color and a memory within the display itself, require another year or two of development. Liquid crystal displays (LCDs) react too slowly to handle rapidly-changing illumination configurations (as would be needed to handle, say, large amounts of descriptive data in an "advancing line" reading mode), normally have no shades of gray and have X-Y-addressing duty-cycle problems. The first pocket computers will probably contain LCD displays, for although they are slow to react, they are cheap and the technology is available now to mass produce them for only a little more than a 12-character line of light-emitting diodes (LEDs)


MAN CREATED TOOLS CREATED MAN CREATED TOOLS CREATED
MAN CREATED TOOLS CREATED
MAN
CREATED
TOOLS
JRL 77

Will the circle remain unbroken?

Marketing Problems

Although the two companies that have introduced magnetic-card-reading programmable calculators consider these devices small general-purpose computers (and technically they are, according to the textbook definition of a computer: an electronic I/O device with a memory that follows a program it can store and responds to co-directional instructions), most of us would include in the definition of a computer the requirements of full alphanumeric character generation and display capabilities, a multiline display and extended sequential and random access. The companies who are most capable of introducing the pocket computer have a vested interest in the status quo, for they all also produce programmable terminals and/or small general-purpose computers and hence are reluctant to add a new competing line, and a lower-priced and lower-profit one at that. Primarily for this reason, special-purpose new-market versions of the pocket computer will probably be introduced first, the initial ones certainly before the end of the decade.

One version would be for computer-assisted instruction. Another would be for crime countermeasures, such as searching a pre-recorded microcassette (updated each morning over the telephone) for a license-plate number, a person's name, a vehicle registration number, a stock certificate, bond or currency number.

In years to come, the sky seems to be the limit. Future general-purpose versions of the pocket computer will allow numerous innovative uses, even the high-speed loading of newspapers, reference data or books at special coaxial loading stations. For more specific and less extensive data needs, eventually (with perfected optical scanning) the entire Library of Congress will be machine-readable and capable of being accessed through any telephone. ■

Microprocessors — A Primer

Theodore J. Cohen, PhD

A sophisticated electronic device known as the microprocessor will shortly have a profound impact on our way of life. Within a year or so, this device, about half the size of a matchbook, will be incorporated in a variety of consumer products ranging from automobiles to digital watches.

What are microprocessors? Why are they important? How do they function? And how will they be used in consumer products? These are the questions answered here.

The Heart of a Computer System

All computers consist of five basic sybsystems:

- An input device through which instructions and data are entered into the computer;
- A central processing unit (CPU) which controls the computer's operation;
- An arithmetic logic unit (ALU) which performs mathematical operations;
- A memory, in which instructions and information are stored;
- An output device, through which processed data leave the computer.

The heart of this basic computer system, which consists of the central processing unit (CPU) and the arithmetic logic unit (ALU), can be incorporated on a single integrated circuit (IC) chip, and this chip is known as a microprocessor.

While early microprocessor-based computers required a considerable number of IC's (30 or more) to recover data from memory, second-generation microprocessors permit the construction of computers having as few as two chips. Thus, it is not unusual to find that the microprocessor is often referred to as a "computer on a chip."

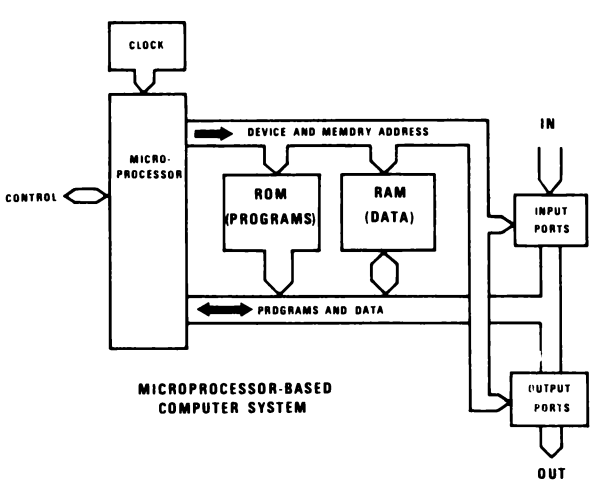
A New Electronics Era

Despite the rapid advances which have been made in electronics since the introduction of the transistor some 30 years ago, many consider the development of the microprocessor as heralding the beginning of a new electronic era. The reasons for this are many. For example, some arithmetic and computational capabilities available in today's microprocessor-based systems would be impractical to duplicate using more conventional circuitry. Then, too, the use of microprocessors results in drastically-reduced product design time, reduced product complexity, and hence, lower product cost. Finally, microprocessor-based products can be programmed to execute a sequence of instructions, and thus can control, or interact with, a variety of instruments, machines, and systems.

In short, the capabilities inherent in microprocessor-based products represent a significant advance in computational and control circuit design.

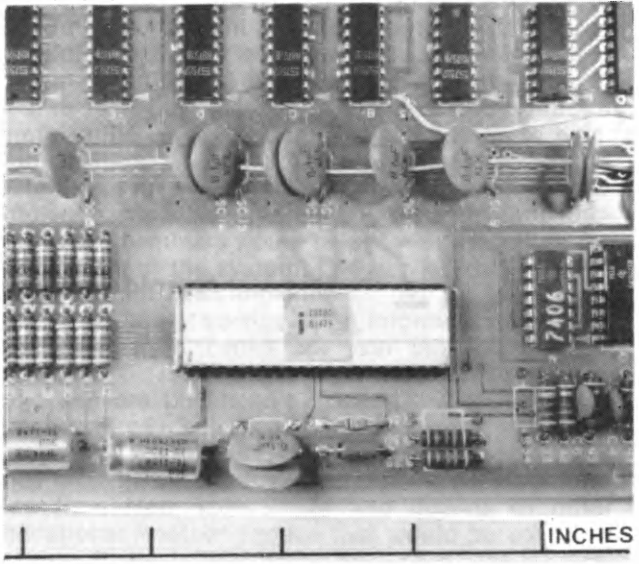
The Microprocessor as a Circuit Element

As already seen, a microprocessor can form the heart of a computing system . . . the heart of a microcomputer, if you will. Here, the microprocessor, together with such additional components as read-only memories (or ROM's, which are used to store the microcomputer program), random-access memories (or RAM's which are used to store data) and interfaces for peripheral devices, is so connected as to perform computations and to make decisions. The microprocessor determines what external devices should provide or have access to data, performs calculations using the data provided, and makes decisions based on these calculations and upon timing constraints which may be imposed by the user. Looked at another way, the microprocessor, which is only one component of a microcomputer, coordinates the activities of the memories and the input-output devices, and



MICROPROCESSOR-BASED COMPUTER SYSTEM

Intel's 8080 microprocessor, a popular second-generation device, contains the equivalent of 5000 transistors as well as most of the basic operational features found in present-day minicomputers. The chip itself is about 1 cm² and 0.1 mm thick. It is mounted on a plastic package called a DIP (Dual In-Line Package) about 5 x 1-1/2 cm. The MPU is dwarfed by the other discrete components (resistors and capacitors) on the PC board.

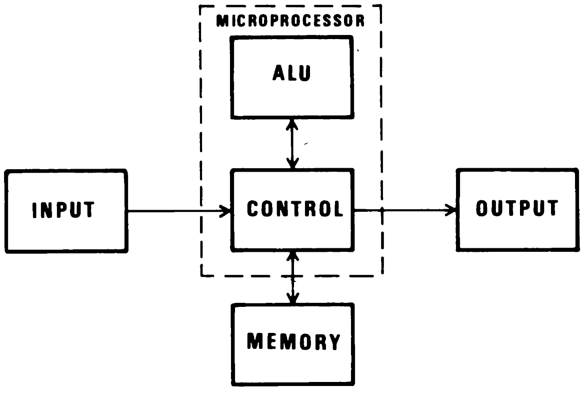


performs logical or arithmetic operations on the data stored in RAM. Used in this manner, the microprocessor makes it possible to incorporate decision-making and data-processing capabilities in a variety of products ranging from automobiles to watches, and from calculators to television receivers.

There's a Microprocessor in Your Future

If you drive an automobile — and most of us do — there's a microprocessor in your future. The need for more dependable, fuel-efficient vehicles makes the automobile a prime candidate for early applications of microprocessor technology. Through the use of an on-board microcomputer, it will soon be possible to monitor such diverse parameters as engine speed, ignition timing, engine temperature, compression, and emission, and to determine automatically that point where fuel economy and emission control are optimized. It will also be possible to determine more accurately when shifting should occur, thereby minimizing transmission damage. Even diagnostic analyses of critical engine functions will give the driver advance warning of impending breakdowns.

While the on-board microcomputer is monitoring your vehicle's performance, it will also be watching out for you, making your ride smoother and safer. Don't worry about your doorlocks; the computer will lock the doors for you once your car's speed exceeds 5 m.p.h. The microprocessor-based computer will also monitor your braking system (to prevent lock-up), and your speed (to warn of excesses). The onboard computer will even be able to provide anti-theft security by disabling the ignition control system when your car is entered without a key having been used.



A BASIC COMPUTER SYSTEM

Now that your appetite is whetted, consider how microprocessors will be used to improve the performance and capabilities of the following products:

Digital Watches

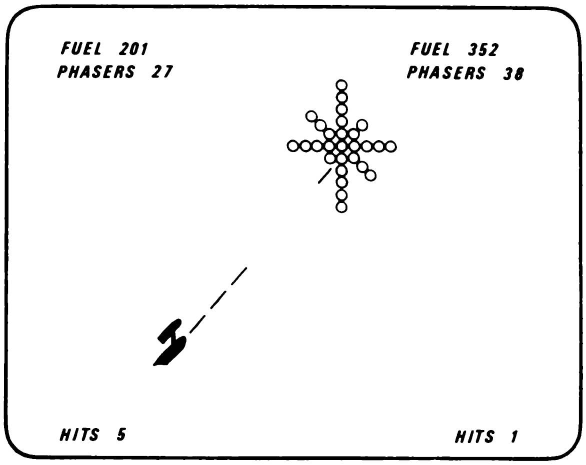
Engineers are already working on microprocessor-based watches that will include a calculator, an alarm, and an interval timer. It may even be possible, someday, to own a watch which provides personal physiological monitoring.

Hand-Held Calculators

While mature, the calculator market is certainly not saturated. Newer more complex units will soon be available, and some may even be able to monitor such body functions as blood pressure or pulse count.

Television Receivers

Microprocessors will permit the expansion of today's television receiver into a comprehensive recreational and entertainment center. Through advanced technology, it will be possible to play a wide variety of games, either against an opponent or against the microcomputer itself. Liberal use of color will make video games more exciting as will the generation of more realistic game sounds. It will even be possible to play games which provide a challenge to players having a wide range of skills; in this way, users will not lose interest as their skills improve.



An Electronic Revolution

A revolution is upon us! Developments in microprocessors are changing the electronics industry at an unprecedented rate. As a result of the changes, a new generation of "smart" consumer products will soon be available . . . products which are not only more capable, but which are also designed to analyze data and to make decisions which permit a variety of tasks to be performed in a highly efficient and dependable manner.

Portions of this article appeared in "Microprocessor Technology — An Electronic Revolution," T. J. Cohen, *Sea Technology,* March 1976. ■

Coin-in-the-Slot Computing at a Public Library

Harold M. Shair *

The public library, a fixture in every community, is a natural location for public-access computing. The concept that a public library is a place where you only take out books has gone the way of the Stanley Steamer. If your library doesn't loan out records and artwork, if it doesn't hold field trips and events, then it's time to throw out the library board. In fact, the modern public library can be thought of as a complete community information and activities center.

At a community information center, it's only natural that public-access computing be available. In a public library, the computer should not have any restrictions on its use. If it's to be used for fun, so be it; if it's to be used for business, that's all right too. Programs of general interest should be made available for people who know nothing about computers. Storage facilities or media should be available for those who wish to write their own personal programs or store their own data. Courses on computer applications and programming should be offered, and events such as contests and fairs should be held.

The first installation to try to achieve these aims is at the White Plains Public Library, in Westchester County, near New York City. In order to place a computer in a public library, several unique conditions had to be met:

1. It had to be installed as a concession, since the capitol budget of the library couldn't stand the cost of an outright purchase.
2. A fee for use had to be charged, not only to pay for the installation, if possible, but even if the money were not needed, to engender respect for the value of the service and equipment.
3. The computer had to be as self-service as possible, since it could demand only minimal support from the library staff.

The computer used in this installation is a Wang 2200B minicomputer with 8K of user memory available. The Wang was chosen because it has a permanent BASIC interpreter and operating system in ROM, a non-menacing typewriter-like keyboard and a 12-inch (diagonal) CRT. Their users group, called "SWAP," was also available as a source of programs in several categories. The programs are stored on cassette tapes, which are kept at the reference desk. Also available at the reference desk is the complete set of reference and programming manuals.

In order to charge for its use as well as provide for the minimal support from the reference staff, the computer is coin-operated. The coin box is a timer, similar to those found in coin laundries, which interrupts the CRT line to blank the screen of the CRT when the time is up. When more coins are inserted, everything is restored as it was. The charge is presently 25 cents for 4¼ minutes ($3.50 per hour).

To use the computer for the first time, a patron has to follow instructions on a wall chart above the computer. At a certain point, the CRT takes over and the programs provide their own operating instructions. The collection of software consists of games and demonstrations, personal finance, educational demonstrations, mathematics, statistics, finance and engineering. The list of programs available (below) is as provided for library patrons. As expected, the most popular use of the computer is game-playing. Bowling tournaments have been held and trophys awarded, a library first. In addition, an ongoing Startrek competition is on cassette. One un-

Wang 2200B minicomputer at White Plains Public Library, with coin-box timer at left.

---

*Consultant to the White Plains Public Library usual program, which unfortunately was banned, would have helped improvers of the breed massage statistics derived from the pages of the *Daily Racing Form* as an aid to investment decisions. The library would not let that program be offered even though the basis of the program came from a book borrowed from the library.

The future for community computers in public libraries is cloudy. Public institutions are lucky these days to maintain what they have and can hardly be expected to invest in "way-out" ideas. The average library director is not familiar with computers and perhaps is a bit afraid of them. A concession would overcome these problems but revenues from use alone are at present insufficient to make it profitable. Additional revenue from courses and seminars would help, as well as add to the pool of users of the system. One service that is feasible, but has not been implemented due to lack of time and money, is self-service retail information-bank access. Information banks are vast bibliographic resources on disks. The two most relevant to general public use are Lockheed's "Dialog," with technical, educational, psychological, business and many other bibliographical abstracts. The other is the New York Times Information Bank with access to 25 million articles from *New York Times* and dozens of other publications. Another service that would be useful in the White Plains Library would be "Lexis," the legal information bank, since the library shares the same plaza as the county and state courthouse. The use of these data banks, however, can run from $50 to $200 per hour on a retail basis, with the average search taking 15 minutes. Those fifteen minutes might replace a week or more of catalog work. Many businesses and universities subscribe to these services, using a terminal for access.

This service could be offered on a self-service basis with local credit-card billing. The library computer would check the credit of the patron and initiate the call to the information bank using an automatic dialing unit. It would perform the necessary handshakes, preprocess data and keep tabs on the customer's bill (even to the point of signing off automatically when a preset time and/or money limit is reached). Training on how to use particular data banks could be provided by programs that "play" information bank, as well as by seminars. There are some libraries now that offer this service, but it is tax-supported either through library or NSF funds.

The role that the new personal computers can play in the future of public-access computing in libraries is also under investigation. ■

> The public library, a fixture in every community, is a natural location for public-access computing.

LIST OF COMPUT-O-MAT PROGRAMS

INTRODUCTION TO COMPUTERS -- Recommended for library patrons who are not familiar with the Comput-O-Mat.

GAMES -- This side of the cassette contains 7 game programs:
1 Horse Race
2 Craps
3 One-armed Bandit
4 Tic-Tac-Toe
5 Blackjack
6 Bowling
7 Football

BRAIN GAMES -- This side of the GAMES cassette contains 5 more advanced game programs:
1 Cryptograms
2 Submarine Commander
3 Arithmetic Quiz
• 4 Stock Market
5 Guess
6 Lunar Lander

•Recommended for adults with some knowledge of basic securities transactions. Game is designed for two or more (up to 10) players.

NEW GAMES -- This cassette contains 6 additional games:
1 Computer Reader & Advisor
2 Flying Saucers (up to 4 players)
3 Space Challenge
4 GHOST -- a word game
5 Calendar
6 HANGMAN -- a word game

Side 2

1 Pizza Delivery Game
2 Biorhythm Analysis
3 Game of Life
4 Wonderama Snake Can Game

CONSUMER FINANCE -- This cassette contains 3 programs:
1 Balance Your Checkbook - This program reconciles a user's checkbook against his bank's statement.

2 Consumer Loans - There are two parts to this program. The first part displays interest rates and payment amounts for mortgage loans, auto loans, home improvement loans, or personal loans available from several banks in the White Plains area. The second part checks an actual loan for compliance with the requirements of the Federal Reserve Regulation Z (Truth in Lending Act).

3 Witholding - This program calculates your Federal, State, and City witholding tax.

HUNTINGTON I -- This cassette contains 7 educational demonstration programs in a variety of fields:
1 DECAY2 - (Physics) -- Solves problems involving decay of radioactive elements.
2 QUADRT - (Math) -- Solves for the roots of a quadratic equation.
3 CLIMAT - (Earth Science) -- A quiz program in climatology.
4 EQUIL1 - (Chemistry) -- Solves problems involving chemical equilibrium of solutions.
5 NZYMC - (Biology) -- Computes enzyme activity as a function of pH, temperature, etc.
6 STOCK - (Social Studies) -- Simulates stock market transactions.
7 AVERG2 -(Teacher Aid) -- Calculates class average grades.

©CREATIVE COMPUTING

RACING FORM

...We'd like to put two bucks on "Happy Daddy" running in the seventh today...

Computer Power to the People!

The myth, the reality, and the challenge

David H. Ahl

The following is a lightly edited transcript of a presentation originally given at the "Man and the Computer" symposium at Dartmouth in December, 1976. Modified versions have also been given at several other educational and hobbyist conferences. Some 80 slides and graphics are used in the live presentation, most of which, unfortunately, cannot be reproduced here.

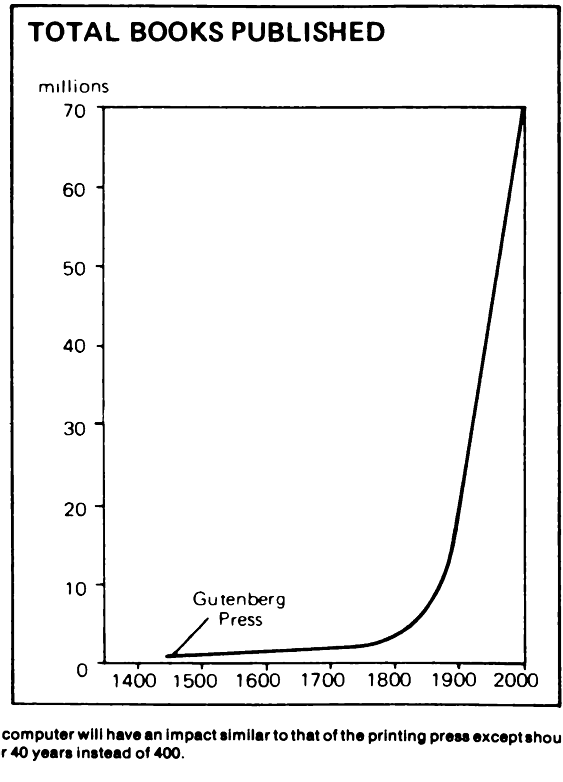
INTRODUCTION

We all know that computers are around us. They're invading our lives along dozens of dimensions. We see them in supermarkets—the little product code you find on the side of virtually every food and grocery product you buy can be read by an optical scanner connected to a computer. Computers in department stores—a little "magic" wand, actually a tiny laser device, reads a product code from the tag. Medical facilities—hospitals frequently keep all their patient records on computers. When you're admitted you often undergo some kind of questioning process. One psychiatric hospital out in Utah takes the entire patient with an on-line computer program. College admissions at, for instance, Fort Lauderdale Community College, and hundreds of others, use on-line computers. Every time you pick up the telephone and dial it you're actually using the largest general-purpose computer in the world—the switched telephone network. Magnetic-ink character recognition in the bank; sports stadium score boards; and so on.

My premise is that now, some 30 years or so after the invention of the computer, it's having a tremendous impact on our lives. It is having an impact on our lives similar to that of the printing press, but instead of taking some 400 years to make its effect known, the computer is having a vast effect in something like 20 or 30 years. We just can't escape it. So some thirty years after the invention of the computer we decided it would be a nice idea to find out what people think about computers. Do they view it as a master, a slave, a dictator, a monster? In fact, do people really understand what the computer is all about and what it's good for? We took a survey among both adults and young people with 17 different questions. We posed statements and asked them "Do you agree with this statement or disagree?', and got their responses. We also had some open-ended questions and we continue to ask people open-ended questions. Like, "if you had a computer in your home, what would you do with it?"

THE MYTH

First of all we asked some questions about what you might call the quality of life. Did people feel that the computer was going to improve various facets of society? For the most part, there was pretty good agreement that computers would improve education somehow, a very substantial agreement that computers would improve law enforcement, a little less agreement, particularly amoung younger people, that computers would improve health care; and some agreement that computers are worthwile for prevention of fraud through credit-rating data. This last one is interesting. The question was asked in the *AFIPS/Time Magazine* survey just four years before this one; the percentage of people that felt credit checking was a good application dropped from 74% to 64%, so 10% more people today have doubts in contrast to four years ago. I guess in four years many people have gotten stung in one way or another by credit ratings or other foulups.



The computer will have an impact similar to that of the printing press except should take 30 or 40 years instead of 400.

Do you feel you can escape the influence of computers?

Influence of Computers

We asked some questions about the threatening nature of computers. Do you feel you can escape the influence of computers? Well, people for the most part felt that they couldn't; a suprising number of young people felt they could. I'm not quite sure where they were going to go to do it, certainly not the United States. There was some feeling, particularly pronounced among West Coast respondents, that the computer could influence the outcome of elections. Senator John Tunney of California was one of the biggest critics of the use of computers to forecast the outcome of elections. Senator Tunney, if you'll recall, was defeated in November, 1976. I'm not sure if computer projections had anything to do with his defeat but, in fact, his fear was that by the time the voters went to the polls in the western states, the major national election would be locked up. In 1976 it wasn't quite locked up by the time they went to the polls, but frequently it is and therefore people may say "why bother" or "gee, there's a bandwagon; I want to get on it and vote for the winner." Or, "I was going to vote for the other guy, and he has lost, so I can't be bothered going to the polls." Well that may not affect the outcome of the national elections, but it has a tremendous affect on the outcome of local elections and local bond issues. So, John Tunney at least was pretty upset about using computers in the forecasting of election results.

"Computers dehumanize society by treating everyone as a number."

"Computers dehumanize society by treating everyone as a number." On that statement we had some ambivalence. Some people agree, some people disagree—certainly a substantial number of people are a little bit fearful and do feel like the computer is dehumanizing by treating them as a number.

The Role of a Computer

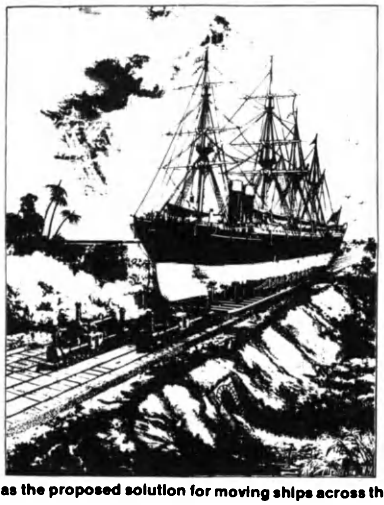
We asked five questions to get at whether people understand the role of a computer. Do they really know what it's good for and do they know its applications? One of those statement was "computers are best suited for doing monotonous, repetitive tasks." Well, 80% of the adults agreed with that, although only 67% of the young people did, which gives rise to the hope that young people can see that computers are good for doing more than just dull, repetitive tasks. Are computers a tool? Yes—a pretty substantial agreement that they are a tool. I think that's a good thing. But I think it matters a lot whether people view it as an intellectual tool or whether they are thinking of it as a plain, ordinary tool such as a hammer, for example.

Do computers slow down and complicate simple business operations? Some people felt that they did—I'm not quite sure who. There's a substantial agreement that computers are going to replace a lot of jobs and create jobs that need specialized training, and some

Statistical Results of Survey of Public Attitudes Towards Computers in Society.

	ADULT (N=300)		YOUTH (N=543)	
	Strongly or Mostly Agree	Strongly or Mostly Disagree	Strongly or Mostly Agree	Strongly or Mostly Disagree
Computer Impact on the Quality of Life				
• Computers will improve education.	86.6%	5.9%	84.2%	4.5%
• Computers will improve law enforcement.	81.9	3.3	70.0	10.1
• Computers will improve health care.	78.6	5.3	54.1	11.9
• Credit rating data banks are a worthwhile use of computers.	64.2	13.4	64.0	7.6
Computer Threat to Society				
• A person today cannot escape the influence of computers.	91.6	4.0	66.6	17.7
• Computer polls and predictions influence the outcome of elections.	48.1	27.5	44.2	26.9
• Computers dehumanize society by treating everyone as a number.	37.4	50.3	39.9	30.6
• Computers isolate people by preventing normal social interactions among users.	18.7	62.5	20.9	42.5
Understanding the Role of Computers				
• Computers are best suited for doing repetitive, monotonous tasks.	80.0	10.3	57.0	21.6
• Computers are a tool just like a hammer or lathe.	72.6	14.7	61.3	23.4
• Computers slow down and complicate simple business operations.	17.6	66.4	17.4	68.8
• Computers will replace low-skill jobs and create jobs needing specialized training.	71.0	15.0	61.8	14.4
• Computers will create as many jobs as they eliminate.	62.5	16.4	40.0	29.1
Understanding of Computers				
• Computers are beyond the understanding of the typical person.	25.2	61.6	30.6	49.2
• Computers make mistakes at least 10% of the time.	9.6	76.7	10.3	60.0
• Programmers and operators make mistakes, but computers are, for the most part, error free.	67.0	19.3	72.3	13.3
• It is possible to design computer systems which protect the privacy of data.	60.2	26.4	48.6	15.9

What would you do if you had a computer at home?



In 1884 this was the proposed solution for moving ships across the Isthmus of Panama.

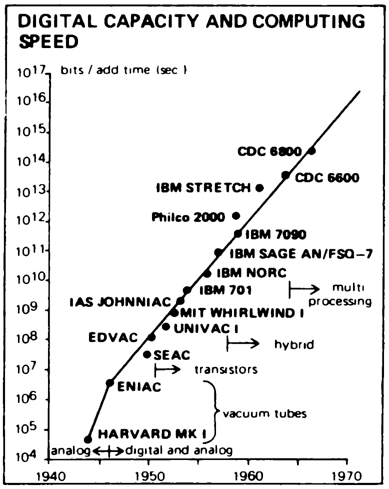
people really fear that they might not be qualified for the jobs that will exist after the "computer revolution." Also on the jobs issue, we asked whether people feel that computers will create as many jobs as they eliminate? About two-thirds agree, but that leaves a fair number that disagree. You have to remember that people have always been fearful of any kind of industrialization or technological breakthrough. The Luddites were anti-technology—to them the industrial revolution meant the machines were going to take all the jobs. Well, it just didn't quite work out that way and I don't really think computers are going to take all the jobs either.

Then we asked a couple of questions to see if people really understand the computer itself. We first asked, are computers beyond the understanding of a typical person? The response was mixed. At least a quarter of the people think that they are beyond their understanding, but I'm encouraged by the larger percentage of people who disagree. "Computers make mistakes at least 10% of the time." You have to feel sorry for the 10% of the people who do think that computers make mistakes this often. In fact it is the programmers and operators who make the mistakes and not the computers. But in these questions we gleaned a little bit of intelligence that someplace between 13 and 19% of the people just actually don't know who's running them. They think the computers are running the people, rather than the other way around. A substantial number of people just didn't know, which is also upsetting. So, there's a substantial portion of our society—at least a third or so—that just doesn't know some of the fundamental issues and facts about computers. We asked one last question—is it possible to design computer systems to protect the privacy of data? Well, not even the computer designers know for sure, so I don't think we could expect much from people that we asked.

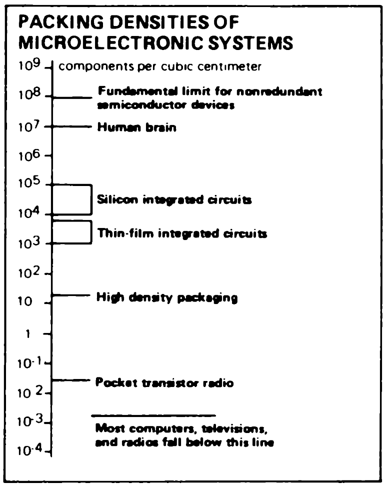
All in all, we have some ambivalence, people optimistic on some counts and pessimistic on some other counts and some things that they just don't know. The ignorance is probably most apparent when you ask someone what would you do if you had a computer at home? A computer? What do you mean a computer? You mean like a hand calculator? Some people thought we meant robots. "Well, maybe I'll have it serve me martinis when I come home from work." They just couldn't quite visualize a computer at home. A computer is supposed to be something that goes behind glass doors and is raised on flooring and requires a lot of electricity. "I don't have the kind of home that would suit a computer," said one.

Everyday Perceptions

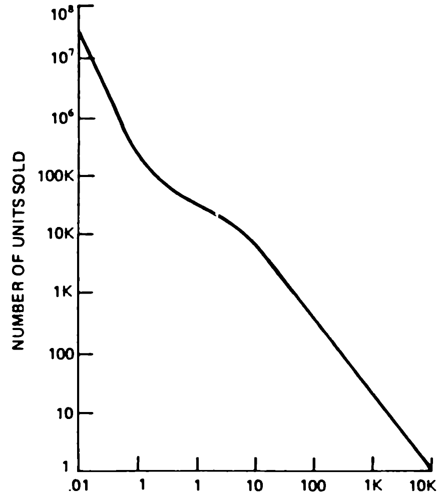
I guess this mixture of attitudes really shouldn't be too surprising. The everyday perceptions of a computer are formed by people in the media and elsewhere who really don't know what computers are all about either. For example, newspapers, comic strips, TV, and so on. What does a newspaper cover? They're going to report the computer error, the problem with the computer. A New Jersey supermarket had brand-new laser scanning systems at the checkout for the grand opening day and they really crammed the people in. Hundreds of people all filled their carts with these grand opening specials. People were lined up at the cash registers, each with two and three carts full of groceries. Seven or eight deep at every cash register and all of a sudden, bang, the system went down. Well, not only did it go down, but it locked all the cash drawers. So there was no way of making change. They couldn't use the cash registers manually. There was just no way of opening them up. Rumors started flying around. People said, "The cash drawers are locked, the doors are going to lock too;



Ever since the first pure digital computer, internal speeds have increased by an order of magnitude (x10) every 2½ years.



In 1953, a computer 'CPM) weighed approximately 5000 pounds, occupied 300-400 cu. ft. and required 40 Kilowatts of power. Compare that to today's microprocessor on a chip!

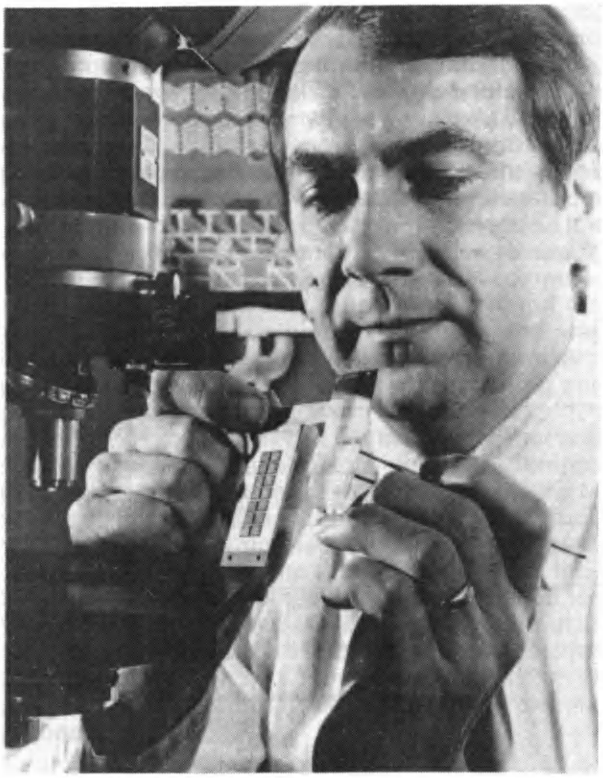


The Free marketplace in action. As prices come down, sales go up.

we're going to be locked in here forever." And then there was a rumor that a replacement computer was going to have to be shipped in from Texas and they'd have to wait until it arrived! It was wild. Finally the manager decided that the best course of action was to give each checker a pencil and some brown paper bags, and have them add up manually the groceries in these laden carts. People were there for *hours.* The interesting thing is they did not lock the doors and more people kept streaming in. The manager didn't want to lock the doors because of this panicky rumor inside the store that if we lock the doors we might be stuck here. They didn't want to start a riot. Well anyway, the newspapers had a field day with the story.

Most of you have heard about the friviality out in Southern California when McDonalds had a sweepstake. To enter, all that was required was a 3 × 5 entry form or facsimile. In other words you could write the entry on a 3 × 5 card of your own. Students at one fraternity programmed the computer to produce entry forms—1.2 million of them—and then they stuffed every McDonalds ballot box in Southern California. They won 90% of the prizes in the contest. McDonalds was very upset about it—they said it was anti-American. I think it was very American; it showed a lot of ingenuity and creativity. In fact, McDonalds awarded duplicate prizes to people that were not members of this conspiracy to defraud them. The winning fraternity invited Ronald McDonald to make the prize presentations over at their fraternity house for dinner, but he declined the invitation. Actually, Burger King got the best publicity out of this. They gave a $3,000 scholarship to the university in memory of the prank. Again, the newspapers had a wonderful time blaming the whole thing on a computer.

A college student at the University of Arizona insured the life of his guppy. He put down all the correct information on the mail order insurance form—height 3 centimeters, weight 30 centigrams and so on. It died of course, as most guppies do, some four or five months later. He submitted a claim for the $5,000 he had insured it for. The insurance company said it was an invalid claim—the computer had made a mistake in accepting this "person." Well the computer hadn't made a mistake—it was a programmer who hadn't allowed for somebody that was 3 centimeters high. It wasn't the computer. But the newspaper, how did they portray it? Sure—another computer error.


This new "bubble" memory developed at Bell Labs can store the information equivalent of 27,000 telephone numbers.

Bell Labs
Bell Labs
Visual communications over ordinary telephone lines is in the works. At Bell Labs a Flat-Screen video device can be used to transmit handwriting instantaneously.

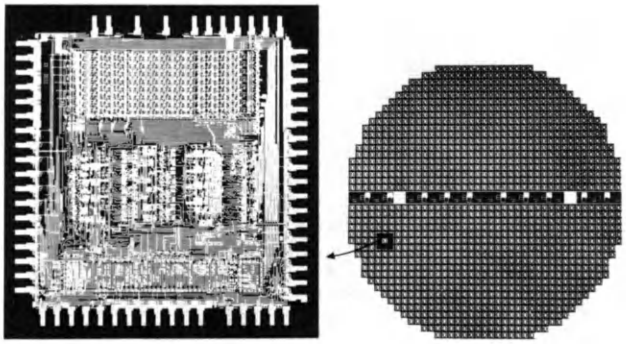
In Swansea, Wales, a young man of 17 applied for a driver's license and passed his test shortly after. But when his license arrived, it bore 12 endorsements for a whole array of driving offenses, plus a 28-day driving suspension. Police proved sympathetic when it was found that "the computer at the license office had run wild. The system has not been operating for long," said an official.

There was a cute little notice printed recently in the *Chicago Tribute.* "A COMPUTERIZED bill had this notice on the bottom: Failure to receive this bill is no excuse for non-payment of the amount shown." Why capitalize "computerized?" Does that mean the computer printed that notice on the bottom of the bill. As if the computer could have made that up out of the blue sky? The computer is the scapegoat for the post office now—that's what's really happening!

A woman in Shreveport, La. got a gas bill for $42,474.58. A customer representative at Arka Gas Co. stated, "The computer went haywire and some of those bills got out." Computer error? Hardly. Good for the newspaper? You bet!

Movies and Books

Movies are another way that people form perceptions of the computer. For example, in *2001*, remember when Commander Bowman finally gains access to the memory banks after Hal has been harrassing him for half the trip and he yanks out the circuits one at a time. Finally, Hal breaks down as Bowman performs the first successful interplanetary lobotomy. The movie *Colossus*—have you seen that one? Colossus "wakes up"



A mask for the Electron Beam Exposure System contains 1304 logic circuits, each of incredible detail. As circuits get smaller, the prospect for a "Dynabook" becomes more real.

and gains sentience very much like the computer did in Heinlein's book, *The Moon Is A Harsh Mistress.* Well, Colossus gains it while it's hooked up to its Russian counterpart. The computers are in charge of the National Defense Systems of both countries and the two computers decide between them that it would be kind of neat if they held the population of both of their countries hostage. A movie that will be coming out shortly, called *Demon Seed,* has a computer in it, Proteus IV (appropriately named) equipped with an ominous blue enforcer arm with which the computer keeps people hostage, mainly Julie Christie in the movie (that probably makes it worth seeing even if you don't like computers). Three movies and three impressions of computers—all false.

Some people get their images of computers from books (not too many because not too many people bother to read books anymore). Science-fiction writers are probably the only writers in the country portraying future computers uses reasonably realistically and making some half-decent speculations. Unfortunately, very few people read science fiction, so we don't have to worry about many people getting a realistic view of computers from that source.

Consequently we know a little bit from the survey what people think about computers and little bit of how these impressions were formed from my rather incomplete discussion of it, but I think you can fill out the missing pieces. We know too that if we ask the average person what would you do if you had access to a computer or if you had a computer in your home he really doesn't have a very good idea. In fact neither do many professionals or manufacturers. The fact is that we're really not very good at forecasting the future. We really can't and never have forecasted future technological innovation or invention very well.

Back around the turn of the century who would have forecasted life today as it actually is? In those days the best guess of what the Panama Canal would be, was a railroad pulling ships across the isthmus. Back in those days it probably seemed reasonable. I'm sure if the Wright Brothers had asked the drivers of ox carts what they would do with an airplane they probably couldn't have given them a very good idea. Henry David Thoreau, one of our leading philosophers commented, when he was told that the telephone would permit people in Maine to talk to people in Texas, "but what does a man in Maine have to say to a man in Texas?"

THE REALITY

It's pretty clear that we can't forecast 70 or 50 or probably even 30 years very well, particularly with a high-technology item such as a computer. So let's just look five to ten years into the future. Even so, we can't foresee exactly when everything is going to occur. We would certainly expect that processor instruction speed would continue to increase very rapidly. Packing density will also continue to increase dramatically. Currently, we are within two orders of magnitude of the human brain. Actually, the theroretical density limit for semiconductor devices is higher than that of the human brain. Currently, bubble memory circuits in Bell Laboratories, about 1 centimenter square, will store about 1.5 million bits.

Coupled with miniaturization, prices are rapidly falling. Let me tell you that more than one manufacturer is a little bit alarmed at the projection of hardware prices approaching zero. The indication is that as the prices come down, the numbers of units sold goes up very dramatically. This applies not only to calculators but to computers as well. What happens as prices come down? What do you think the value of this ratio is today?

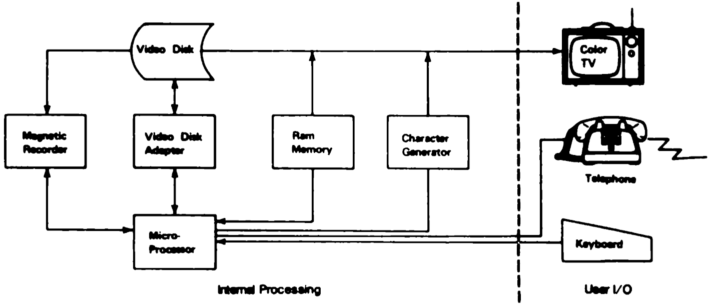
$$\frac{\text{Cost to program 1 line of code}}{\text{Cost to execute 1 line of code}}$$

One hundred to one? A thousand to one? ten thousand to one? Wrong. IBM says the ratio is 100 million to one, and that was two years ago! Given the current increases in processor speed, it's probably a lot more than that today. What that indicates, of course, is that the human element is by far and away the most important thing in computers and technology today, in making them all work.

So what does all this mean when you put all this technology together? Well, obviously it means smaller terminals, terminals that fit in your pocket. Sophisticated and very small color video cameras. Calculators with as much power as a computer of 20 years ago. Hobbyist computer kits that are within the price range of a quarter of the households in the U.S. Close to 30,000 hobbyist computer kits have been sold as of the end of 1976. Technology means people talking to other computers and terminals by means of the telephone network, using standard Teletype terminals or new high-speed terminals or plasma panels built into your phone. A panel that can be written on with a light pen or typed on; or display information from a computer, data bank, directory, or from local storage.

Personal Computers

Today there are over 100 manufacturers of personal computers and peripherals. At *Creative Computing* we



Future Home Computer System

Consumer electronics manufacturers are currently evaluating systems like this for the home.

can't possibly keep up with all the new-product announcements for new hobbyist computer kits and peripherals. We started a new-product section in early 1975 and the hardware portion was about one page. In the Jan-Feb 1977 issue it ran 9 pages of closely-spaced descriptions of new hardware. It's a revolution. Two hundred computer stores open now and a new one opening every four days. Retail computer stores where you go buy yourself a microprocessor, a computer kit, or peripherals.

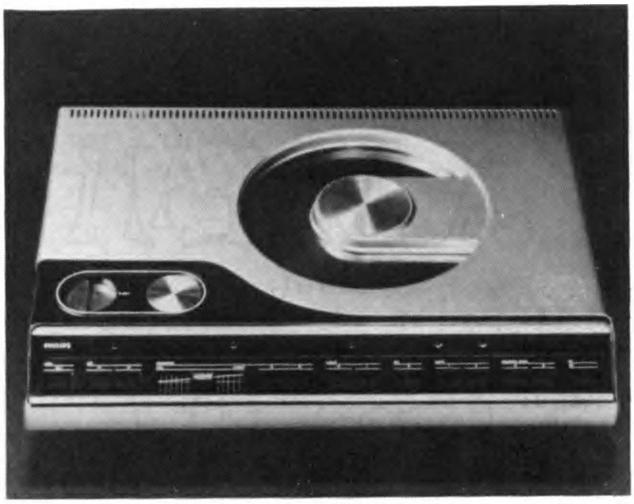
Extensible, user-defined, simple languages are being developed. Harvard has a new language called ECL. It's not like today's simple languages, say BASIC or LOGO. ECL doesn't use constructs that have to be absorbed into your intuition but rather you use concepts that are already part of your intuition, part of your language, and then you construct the computer language out of that. Whether you're a banker, a baker or a professional programmer, you can produce a computer language that does exactly what you want.

Dynabook

Out at Xerox's Palo Alto Research Center they've got a thing called the "Dynabook." The original idea was that Dynabook should in every way be better than a book. It can display printed pages on its screen: black on white, white on black, red on green, etc. it can display pages in any style and size of typeface. If you have some visual problems and you want a page in large type, Dynabook displays it in large type or for reference material it can use very small type. Not only can you read things but you can write things on it. You can just draw a circle around a word and move it to someplace else with a little arrow and the computer moves it for you. You can edit your material from a keyboard if you'd rather. You can strike-over lines and they disappear. The next time you push a button you get all your text nicely justified on the screen. Actually, it's better than a book in every way because you read it, you can write it, and you can change it. It's also better from the graphical sense. It would be nice if the illustrations in a book could move with full animation. In fact not only can they move the way that they're programmed to move, but if you'd like them to move in some special way, you simply take your light pen and draw over the illustration and let the computer sequence through your frames. This is reality. This is here today. It's not quite the size of a book today; it's about the size of three bread boxes but it's not going to be too long before it's the size of a book. In addition to having book qualities, it's also a general-purpose computer with the ability to do parallel processing on eight different levels. When you think of it, that's the way human beings think. When I'm walking along, for example, one part of my brain is thinking about putting one foot in front of the other, another part is thinking "it's cold out, I'll be glad when I can get inside, another part is thinking about the speech I'm going to give tonight, another part is thinking about the person I'm talking to and still another is thinking about the beer that I'll have later on and so on. So your brain is processing information on a parallel basis all the time. Well, wouldn't it be nice if you could have a computer that could do that too and have the output of one level serve as the input to another. That's precisely what Dynabook provides. It's a phenomenal machine. I'd like to think that within 10 years it will be as commonplace as the pocket calculator is today.

Video Disc

I feel one of the keys for getting computers into the home at least is the widespread availability of cheap high-quality software. One possible vehicle for bringing this about is the video disc.



Videodisk players should cost around $500. Each disc can store 10^{11} bits of information in binary format.

Quite inadvertently, a stroke of luck perhaps, the storage technique employed by the disc is binary, or digital. Just what's needed for a computer. So while the player will be brought into the home for entertainment, its real power lies in the fact that if you couple the videodisc system with a microprocessor and keyboard you have incredibly powerful audio/visual/computational/educational/recreational device. One videodisc can store 10^{11} bits of information, the entire Encyclopedia Britannica for example, or a very comprehensive software library. You could have Jackie Stewart introducing the Monaco Grand Prix, taking you on a pre-recorded ride around the circuit, and then turning the controls over to you. Or Kirk handing you the controls of the Enterprise just as the Klingons are about to attack. Or Fran Tarkington coming off the field and putting you in as quarterback in the Super Bowl.

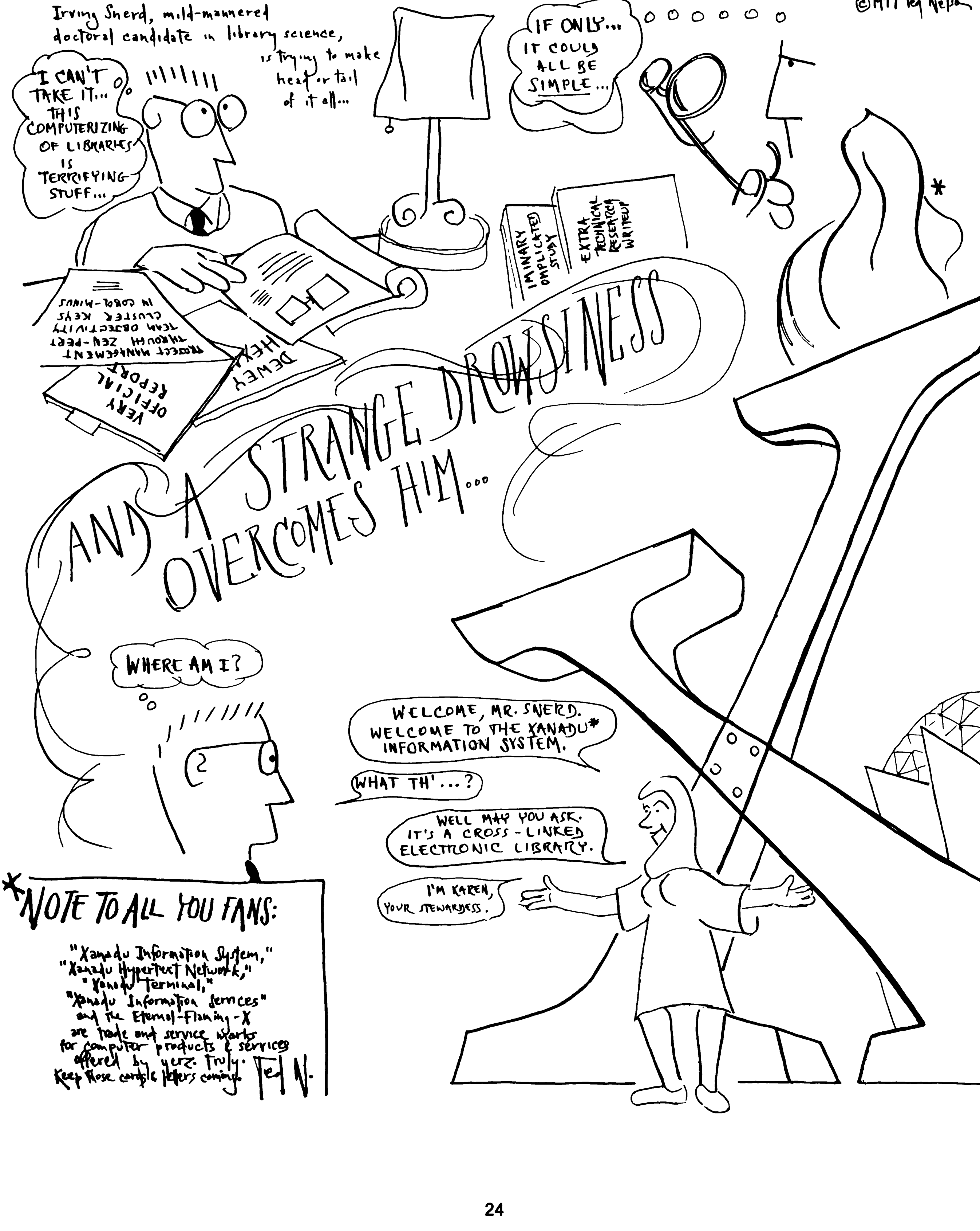
THE CHALLENGE

There's no question that in five to ten years, solid-state and related technologies are going to put some fantastic things well within the reach of everyone who want them. It's equally clear that most people have little idea of what they'd do with a computer if they had one. Hence, we have quite a chasm between the insiders (those who have learned about computers from school, work, or hobby) and the outsiders who don't know much about computers and don't really care (today).

It would be nice to think that this chasm could be bridged by education (like the new math or metric system?), but it's not likely that schools will really face up to computers until every kid has his own (pocket calculators all over again). Business and industry are so wedded to large EDP Systems, with most DP Managers pretending that microcomputers are just toys, that we can't expect any help from that quarter. Most likely it will be the people, plain ordinary folks, who see a friend with a computer and decide to get one of their own., And as this increasingly happens, we're going to have the most massive domino effect you ever saw—calculators and CB move over—you ain't seen nothing' yet. Computer power to the people is on the way! ■

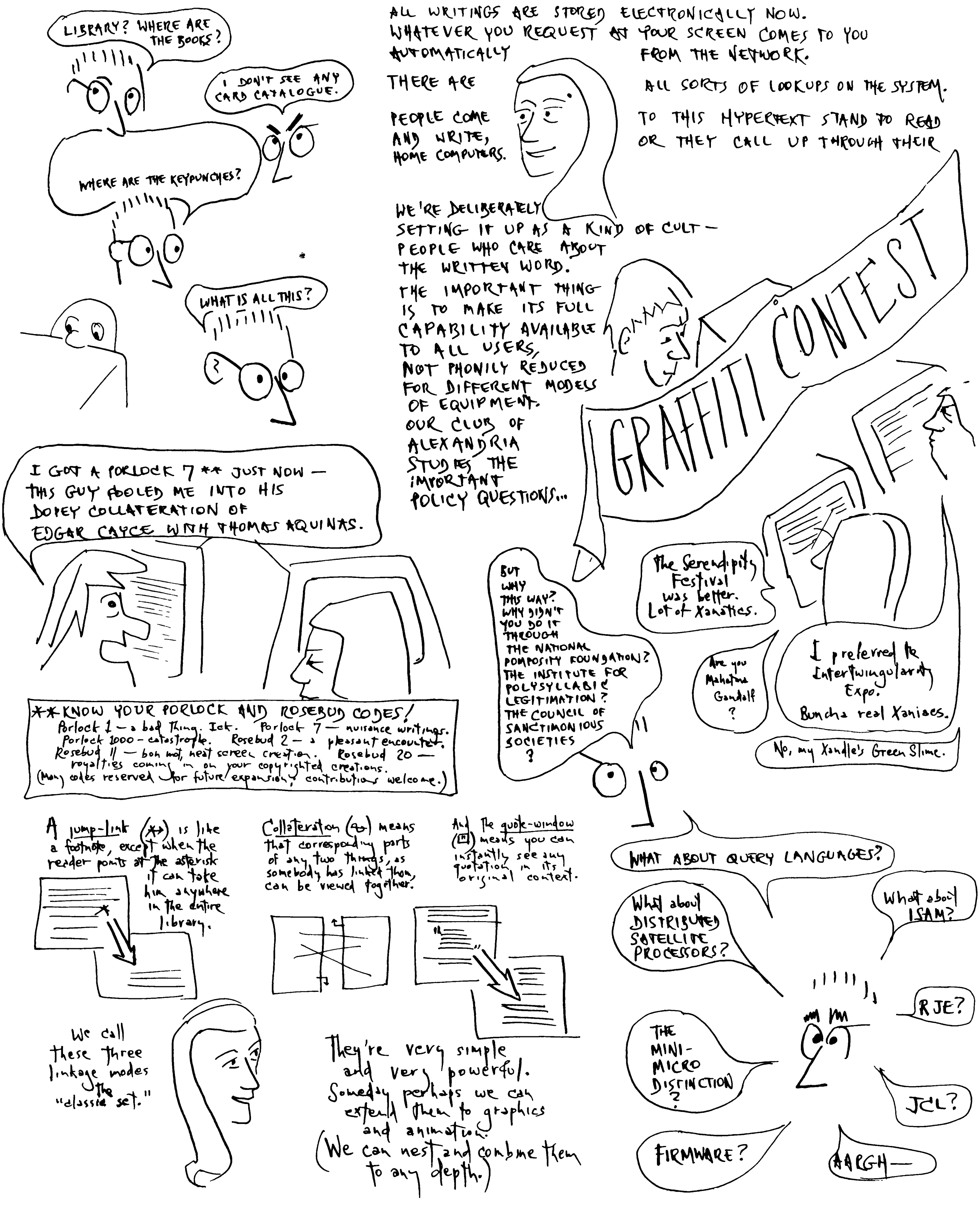
A DREAM FOR IRVING SNERD

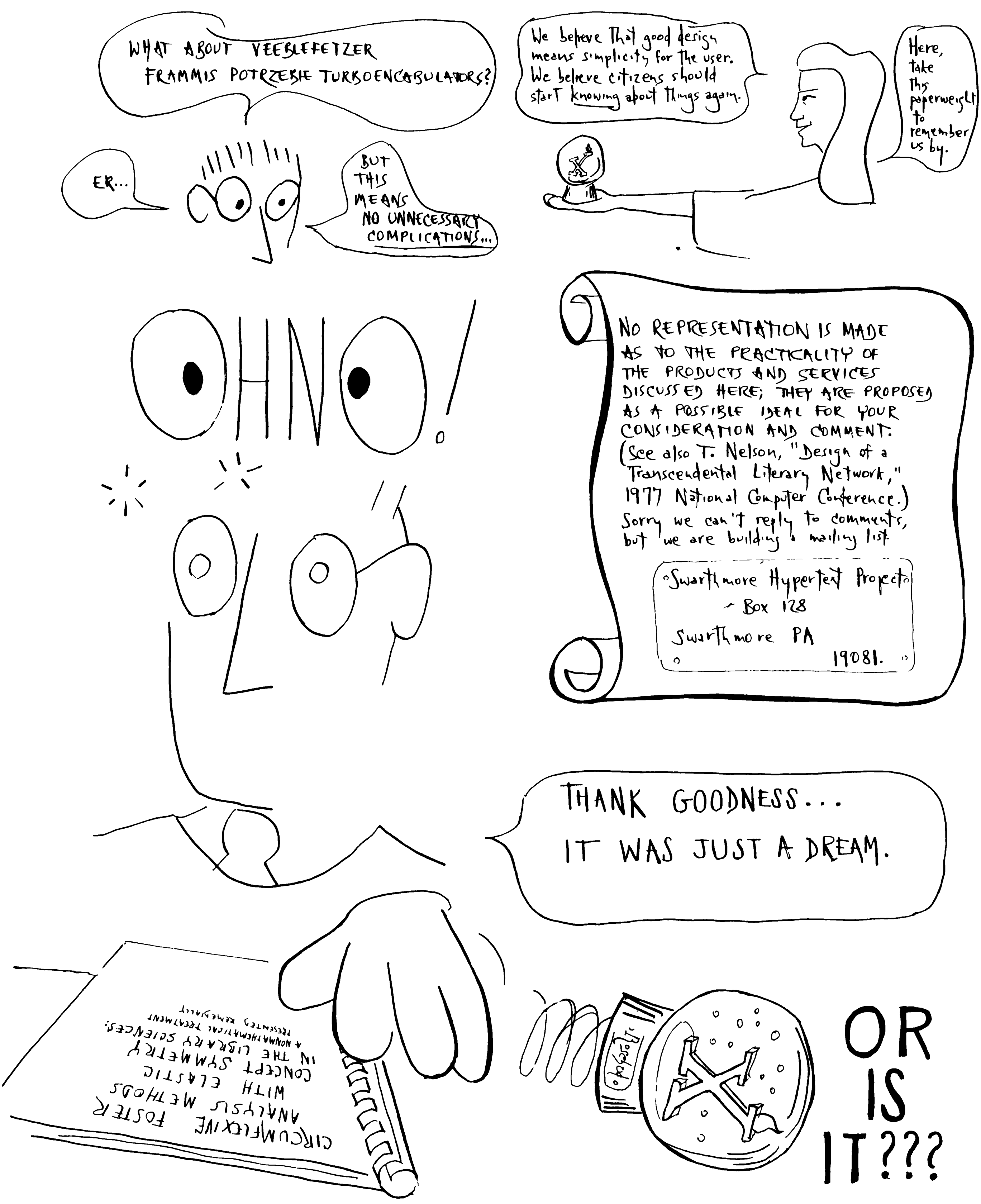
©1977 Ted Nelson


Irving Snerd, mild-mannered doctoral candidate in library science, is trying to make head or tail of it all...
I CAN'T TAKE IT... THIS COMPUTERIZING OF LIBRARIES IS TERRIFYING STUFF...
IF ONLY... IT COULD ALL BE SIMPLE...
PROJECT MANAGEMENT THROUGH ZEN-PERT TEAM OBJECTIVITY CLUSTER KEYS IN COBOL-MINUS
VERY OFFICIAL REPORT
DEWEY HEX
IMINARY OMPLICATED STUDY
EXTRA TECHNICAL RESEARCH WRITEUP
AND A STRANGE DROWSINESS OVERCOMES HIM...
WHERE AM I?
WELCOME, MR. SNERD. WELCOME TO THE XANADU* INFORMATION SYSTEM.
WHAT TH'...?
WELL MAY YOU ASK. IT'S A CROSS-LINKED ELECTRONIC LIBRARY.
I'M KAREN, YOUR STEWARDESS.

*NOTE TO ALL YOU FANS:

"Xanadu Information System,"
"Xanadu Hypertext Network,"
"Xanadu Terminal,"
"Xanadu Information Services"
and the Eternal-Flaming-X
are trade and service marks
for computer products & services
offered by yerz. truly.
Keep those cards & letters coming. Ted N.


LIBRARY? WHERE ARE THE BOOKS?
I DON'T SEE ANY CARD CATALOGUE.
WHERE ARE THE KEYPUNCHES?
WHAT IS ALL THIS?
ALL WRITINGS ARE STORED ELECTRONICALLY NOW. WHATEVER YOU REQUEST AT YOUR SCREEN COMES TO YOU AUTOMATICALLY FROM THE NETWORK.
THERE ARE ALL SORTS OF LOOKUPS ON THE SYSTEM.
PEOPLE COME TO THIS HYPERTEXT STAND TO READ AND WRITE, OR THEY CALL UP THROUGH THEIR HOME COMPUTERS.
WE'RE DELIBERATELY SETTING IT UP AS A KIND OF CULT — PEOPLE WHO CARE ABOUT THE WRITTEN WORD.
THE IMPORTANT THING IS TO MAKE ITS FULL CAPABILITY AVAILABLE TO ALL USERS, NOT PHONILY REDUCED FOR DIFFERENT MODELS OF EQUIPMENT.
OUR CLUB OF ALEXANDRIA STUDIES THE IMPORTANT POLICY QUESTIONS...
GRAFFITI CONTEST
I GOT A PORLOCK 7 ** JUST NOW — THIS GUY FOOLED ME INTO HIS DOPEY COLLATERATION OF EDGAR CAYCE WITH THOMAS AQUINAS.
**KNOW YOUR PORLOCK AND ROSEBUD CODES!
Porlock 1 — a bad thing. Ick. Porlock 7 — nuisance writings. Porlock 1000 — catastrophe. Rosebud 2 — a pleasant encounter. Rosebud 11 — bon mot, neat screen creation. Rosebud 20 — royalties coming in on your copyrighted creations.
(Many codes reserved for future expansion; contributions welcome.)
BUT WHY THIS WAY? WHY DIDN'T YOU DO IT THROUGH THE NATIONAL POMPOSITY FOUNDATION? THE INSTITUTE FOR POLYSYLLABIC LEGITIMATION? THE COUNCIL OF SANCTIMONIOUS SOCIETIES ?
The Serendipity Festival was better. Lot of Xanatics.
Are you Mahatma Gandalf ?
I preferred the Intertwingularity Expo. Buncha real Xanises.
No, my Xandle's Green Slime.
A jump-link (*→) is like a footnote, except when the reader points at the asterisk it can take him anywhere in the entire library.
Collateration (⇆) means that corresponding parts of any two things, as somebody has linked them, can be viewed together.
And the quote-window (☐) means you can instantly see any quotation in its original context.
We call these three linkage modes the "classic set."
They're very simple and very powerful. Someday perhaps we can extend them to graphics and animation. (We can nest and combine them to any depth.)
WHAT ABOUT QUERY LANGUAGES?
What about DISTRIBUTED SATELLITE PROCESSORS?
What about ISAM?
RJE?
THE MINI-MICRO DISTINCTION ?
JCL?
FIRMWARE?
AARGH—


WHAT ABOUT VEEBLEFETZER FRAMMIS POTRZEBIE TURBOENCABULATORS?
ER...
BUT THIS MEANS NO UNNECESSARY COMPLICATIONS...
We believe that good design means simplicity for the user. We believe citizens should start knowing about things again.
Here, take this paperweight to remember us by.
OH NO!
THANK GOODNESS... IT WAS JUST A DREAM.
CIRCUMFLEXIVE FOSTER ANALYSIS METHODS WITH ELASTIC CONCEPT SYMMETRY IN THE LIBRARY SCIENCES: A NONMATHEMATICAL TREATMENT PRESENTED REMEDIALLY
OR IS IT???

NO REPRESENTATION IS MADE AS TO THE PRACTICALITY OF THE PRODUCTS AND SERVICES DISCUSSED HERE; THEY ARE PROPOSED AS A POSSIBLE IDEAL FOR YOUR CONSIDERATION AND COMMENT. (See also T. Nelson, "Design of a Transcendental Literary Network," 1977 National Computer Conference.) Sorry we can't reply to comments, but we are building a mailing list.

Swarthmore Hypertext Project
Box 128
Swarthmore PA
19081.

A national computer club? Will it benefit the hobbyist? What could it, should *it, undertake to accomplish? The chairperson at the First National Computer Club Congress at Dallas gives an insider's accounting on what may become an important organization.*

It's Time For A National Computer Club

Rich Kuzmack

The first National Computer Club Congress was held in Dallas this past June in conjunction with the 1977 National Computer Conference (NCC). About thirty computer clubs across the country were represented by over forty-five club leaders and delegates, who assembled to exchange ideas and discuss issues related to club activities and programs, with great attention focused on the question of a national organization.

Club leaders have been meeting at computerfests for informal discussions since the first fest in Trenton in the spring of 1976, but the prestige of the NCC and the support of Dr. Portia Isaacson as chairperson of the 1977 NCC provided both a tremendous opportunity and a serious challenge to take action toward forming a national organization. Rather than leave a worthwhile outcome to chance, club leaders held pre-NCC planning sessions last spring at the computerfests in San Francisco, Trenton, and Cleveland. In addition, a survey of club leaders was conducted by mail to solicit opinions, especially of those unable to make it to any of the planning sessions or to the Congress itself.

The Congress had two sessions on successive days. The first session addressed the purposes a national organization might serve and the organizational structures that would be appropriate to achieve those purposes. It began with a report on the results of the survey and included a panel discussion on the organizational structures used in the regional organizations of computer clubs: Southern California (SCCS), Midwest Alliance (MACC), and Chesapeake (CMC). (A new regional group was formed at the Congress, the Southwest Federation of Computer Clubs.) A solid concensus of the Congress favoring the establishment of a national organization developed, but no firm concept emerged on the essential characteristics and purposes such an organization should possess.

By the second session of the Congress it was quite apparent that a lot more work would be needed before an organization could be established, but also, that it was time to get started. Accordingly, after considering several proposals it was decided that a working group to be known simply as the Committee would take on the job "to define and establish a national amateur computer society." Membership on the Committee is open to anyone seriously interested in working to achieve its objectives.

The Committee met immediately following the Congress, selected a Secretary to consolidate and distribute communications among its members, and received pledges of financial support from the established regional organizations to cover the Secretary's

Photos by Tom Woodward of SCCS Interface

Big talk in Texas: the Committee meets in Dallas after the first National Computer Club Congress

Unlike other organizations that are formed from the top down, local amateur computing clubs existed in the hundreds before thought was given to the need for going national. Pictured here are three club meetings.

needs now is

At the Chesapeake Microcomputer Club, manufacturers are often invited to discuss their products.

expenses. The Secretary is Jim White (1202 River View Lane, Watertown, WI 53094) of the Durant Computer Club. Each member of the Committee will be setting down ideas, for distribution through Jim to other Committee members, and will be seeking inputs from members of their respective clubs. In addition, articles will be prepared for the various publications that serve the amateur computing community to keep everyone informed and encourage participation in the effort. Finally, members of the Committee will take advantage of computerfests to meet, under a rotating presiding officer, in open forums to which all interested individuals are invited.

Both the Congress and the Committee meeting were able to operate by concensus, essentially deferring those topics on which there was not a broad base of agreement. These issues do have to be addressed and at least tentatively resolved if the Committee is going to make progress toward its objective. In discussing some of these issues I will be suggesting solutions which reflect my own personal opinion. All concerned hobbyist views are needed, and they are encouraged to write them down in a letter to the Committee's Secretary and participate in the formation of a truly representative hobbyist society.

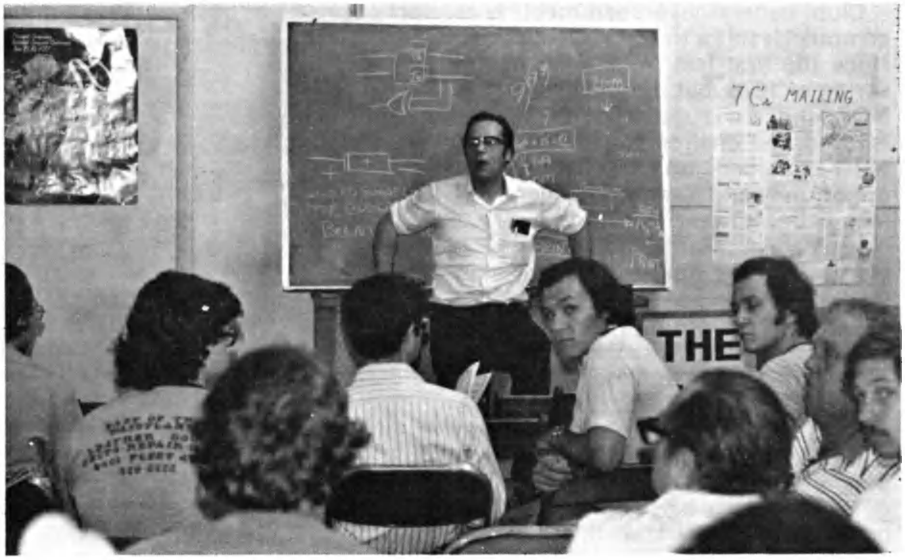
Amateur Defined

"... to define and establish a national *amateur* computer society" is the way the Committee charter reads, but there has been some concern expressed about the term "amateur" by professionals in various aspects of the computer industry. Computer professionals were in the vanguard as the movement to personally-owned computers grew, but it is rare for any one individual to be professionally involved in the breadth of hardware and software, electronic and mechanical devices, and systems and applications programming, with which the amateur computer enthusiast must be concerned. By and large, amateur computing is not unprofessional, but at the same time it should be non-professional. The distinction between amateur and professional, it seems to me, should be based on whether or not the *computer aspects* of an activity or project are being done *for pay or commercial gain.*

Many amateurs are finding interesting work-related applications for their computers. This includes those into computers professionally who are using their own computers for professional development in their field. If I, as an economist, use my own computer to do some statistical analyses for work, that should still be considered amateur computing. A professional systems programmer who writes a routine to drive a video display for his own computer is also doing amateur computing. But the work that the systems programmer does for his employer or his clients should be considered professional and should meet professional standards for quality. And should I offer my computer skills or products for sale, then for that project I would not be an amateur and should be expected to meet whatever professional standards the marketplace imposes.

Questions of Purpose

The next step toward defining a national amateur computing society is



President Lou O'Block tries to silence heckling by John Coklet at meeting of Cleveland Digital Group.

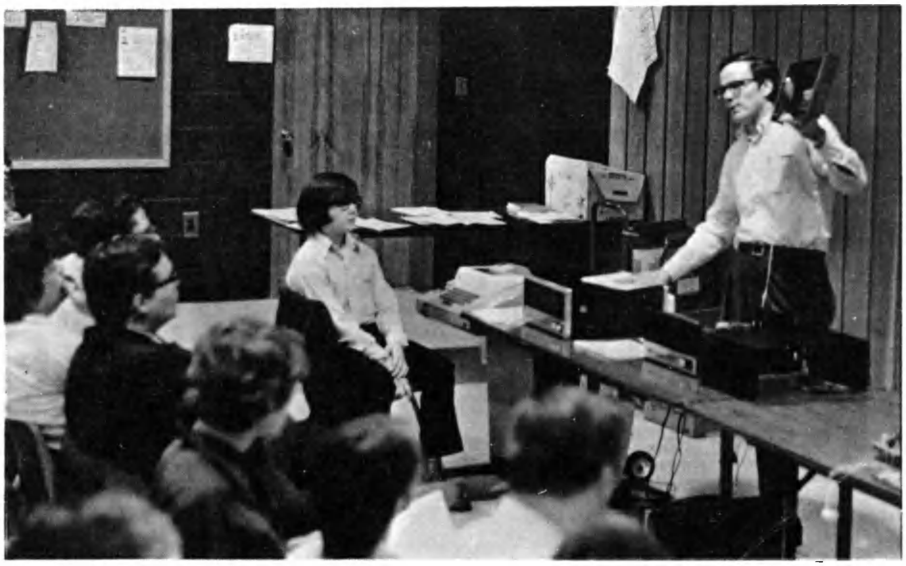
deciding on the purposes the society should serve. The basic purpose, of course, is to serve the amateur computing community, which simply raises the obvious question, "How?" Included among the many good and sufficient purposes appropriate to a national society are standards development and dissemination, computerfest schedule coordination, hardware and software products evaluation, educational and club program services, ombudsman services, and more. There are also some potential purposes that stir up considerable controversy, such as membership directories, group purchasing, and lobbyist activities.

What's the fuss about these last few? Many people at the Congress thought they were pretty good ideas, and taken at face value they seemed worthwhile. The discussion, however, brought out some good reasons against doing them. A membership directory, for example, could really help in getting people together; unfortunately, it would also serve as a handy guide for thieves to some pretty valuable loot. Group purchasing is tempting, indeed, until all the benefits, costs, and alternatives are considered, and then it doesn't fare too well. It would be an administrative nightmare unless run like a commercial mail-order operation, with paid employees, office space, and so forth. The costs for all this overhead would have to be added to the price of the items being purchased. A local or regional club might not get quite the quantity discounts a national operation could muster, but a local club uses the donated time of a volunteer doing it at home. That's a hard act to beat, price-wise.

But this still leaves too many important tasks that could be done best by a national organization, which raises the issue: of which ones should be selected for openers. Clearly, it would be much better to select a few of the most significant ones and do them well than to botch a lot of good ideas by taking on too many.

My personal choices for openers would be hardware and software products evaluation and ombudsman services, with a supporting role in a wider effort to achieve computerfest schedule coordination. I prefer these over other possibilities because they are sorely needed, they are particularly appropriate for a single national organization, and they are outward-looking in that they address the amateurs' interaction with the industry. Many of the other aspects desirable in a national organization will evolve naturally and others can be developed as adequate capability is achieved.

There are at least two successful models on which to pattern a product evaluation function: the Consumers Union and the Underwriters' Laboratory. The primary difference lies in the source of their financial support, with Consumers Union financed by its members and publications, while Underwriters' Lab is industry-supported. Both do laboratory testing of products, although Consumers Union must also depend on surveys of users. For both, the independence and integrity of their work is an essential ingredient in their effectiveness.


John and Wayne Loofbourrow talk to the New Jersey Amateur Computer Club

I think that both amateurs and the industry would be willing to provide the necessary support for a comparable service in the small computer field, although either group alone would probably be sufficient. An ombudsman service could fit in neatly, brokering complaints and remedies more efficiently than the separate parties are able to do individually. Should it be decided to pursue these objectives, the next step would seem to be a survey of prospective participants to ascertain interest and support.

Finally, computerfest scheduling is shamefully uncoordinated despite widespread recognition of the problem. The amateur is distressed by the conflicts that arise, but the exhibitor is really put in a bind to attend the too many shows, some overlapping, with the time, personnel and money they require. While all of us share in the problems, this is one where I believe the companies that are exhibiting will have to take the lead. From the discussions at the Congress I think it's fair to say that the amateurs and their clubs will support any reasonable effort at coordination in this area.

A Matter of Form

Defining a national amateur computer society will naturally have to include some decisions about the form or structure of the new society. There has been a lot of discussion about the basic unit of membership. Should the society be an organization of clubs or an organization of individuals? This is still an open question, but I sense that the weight of opinion is gathering in favor of an organization of individuals. For one thing, a few of the clubs represented at the Congress pointed out that because they were part of a school or company, the club itself could not join although its members could as individuals. There was also concern for the individuals too dispersed to have more than a one-person club. Most influential, perhaps, was the fact that many other national organizations are based on individual memberships with local or regional chapters providing accessible activities.

Unlike other organizations, however, we already have local clubs and regional organizations of clubs, and are only now getting around to forming a national organization. In a sense, the structure of the national society has actually been established and is just waiting to be discovered and incorporated in the definition being developed. An interesting idea proposed at the Congress is that there should be classes of membership which would allow for both individual and club memberships. That would also allow companies to join as sponsoring or institutional members and provide a means for them to support independent product testing.

Three factors suggest that different classes of membership hold the solution to the question of form. First, whatever is decided will have to fit with the organizations that already exist. Second, it will have to be compatible with the functions or purposes to be performed. And third, it must be flexible enough to be able to adjust to new situations and new functions that haven't been thought of yet. ■

The Microcomputer Inflicts "Future Shock" on Technical Education

Richard Vuillequez

The microprocessor is forcing technical educators to reappraise the traditional methods for teaching digital electronics and computer programming due to the convergence in course content. The "computer-on-a-chip" has created a demand for new teaching aids and texts to satisfy people of all ages and experience levels who want to understand the computer but have neither the time nor desire to master all the formal engineering prerequisite courses.

Educators have become increasingly critical of the traditional linear approach to teaching computers, where the student must progress through a number of theoretical courses on devices and analog circuits before being introduced to digital technology and the fundamental logic elements of the computer. Especially since this appoach forces programming itself to be considered an independent subject.

This time-consuming approach may be excellent preparation for the student planning a career in electronic design, but for many other students with specific academic interests and career goals and a desire to utilize the computer for their purposes, it delays computer comprehension and utilization until well along in the curriculum. Students are frustrated by—and critical of—the traditional methods of teaching this evolutionary tool and report that after thousands of dollars of schooling they find themselves unable to use the computer.

Some professors report that even their ablest engineering students have trouble "tying together" their background knowledge in hardware and software to make effective use of the microcomputer in actual system design work, so rapid have been the changes in technology. The ablest students in engineering and computer science undergo the feelings of "future shock."

Editor's note... Richard Vuilleguez is from Derby, Ct. and works for E&L Instruments. In the past Creative *hasn't run articles by manufacturers, but Richard's article discusses an important point in shaping future trends of public access to computing and we felt it belonged in this issue.*

Many people are now fulfilling their desire to use the computer with informal seminars that bypass much much of the material being offered in the rigid traditional courses.

The special microcomputer seminars have seized the initiative and have married the essentials of digital logic, computer architecture and programming into one unified course suitable for everyone from beginning hobbyists to seasoned professional circuit designers seeking an update in the new technology.

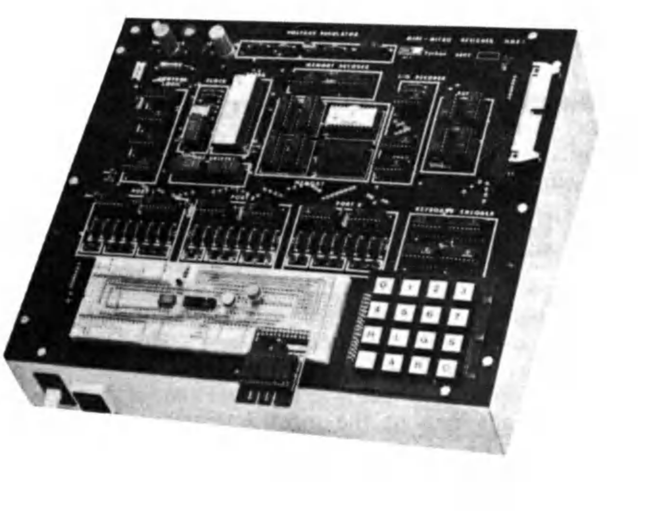
The focal point of this revolution in education is the self-contained, desk-top microcomputer built around a popular microprocessor central processing unit (CPU) chip and a matched set of interface, memory and control chips. They may be sold factory-built or they may be assembled and wired from kits by instructors or students.

A completely assembled unit with keyboard, status lamps, power supply and a reasonable amount of read/write and read-only memory will typically sell for less than $500. Some have provision for breadboarding for interface experimentation. The student will be able to write and carry out simple programs to solve mathematical problems or even control motors, relays or lamps. The trainer is a simple, yet functionally complete computer that is easier to comprehend than a minicomputer and gives the student complete control over both hardware and software.

Crucial issues in the selection of these training aids are the quality and educational level of the accompanying instructional text and the provisions for "hands on" experience in interfacing the microcomputer with external system components. Some trainers are "closed" systems, essentially limiting computational results to a lamp display. Some are also accompanied by manuals or handbooks largely devoted to the internal workings of the chips and incomprehensible except to those with current knowledge of large-scale integration device/specifications.

Professional educators favor the systems that can be employed both as classroom instruction aids for demonstration purposes and for self-instruction where existing curriculums do not permit formal instruction.

The microprocessor is forcing technical educators to reappraise the traditional methods for teaching digital electronics and computer programming.



E & L Instruments MMD-1 Mini-Micro Designer microcomputer, with octal keyboard and breadboarding area.

They emphasize systems that give a student an opportunity to gain an over-all appreciation of the microprocessor and microcomputer with little or no tutorial help other than the texts supplied.

Microcomputer trainers are turning up at all levels of education from high school and vocational school to graduate school. They are being used as demonstrators in formal lectures, as bench equipment in computer science and electronics laboratory courses and as the central hardware in informal two- to five-day accelerated "crash" courses sponsored by professional societies, semiconductor manufacturers and distributors and the educational systems makers themselves.

Microcomputer trainers are turning up at all levels of education from high school and vocational school to graduate school.

E & L Instruments is one of the equipment manufacturers that has responded to the educational crisis brought on by the onrush of the microprocessor into contemporary technology.

They are forerunners of a "hands on" approach to learning computing and are themselves evolving techniques that support this approach. The "hands on" approach makes drastic gains as its students realize it is successful where traditional courses have failed. E & L evolved two teaching techniques, first with texts that were self-instructive to the training kits, then intensive seminars that followed the guidelines of the texts but offered help and encouragement that a beginner might need, lacking confidence to learn to use a microprocessor by a book alone, and that an advanced user could utilize to increase his or her programming sophistication.

It offers the "Bugbooks" that can be used for effective primary training in digital electronics for persons lacking a formal background in electronics engineering, with a series covering logic and memory experiments using TTL integrated circuits, the universal asynchronous receiver transmitter and microcomputer interfacing. The latest series of 'Bugbooks" integrates the subjects of digital electronics, microcomputer interfacing, and microcomputer programming into a single unified course. This approach in itself is innovative, especially in view of the fact that the books are self-instructional.

The seminars are usually two to five days in length and generally have as their objectives:

1. The introduction of the student to the concept of a software-based electronic circuit through actual "hands on" experience with a well-know MPU chip set:
2. The attainment of a comprehension level of the language and literature of computers and programming that will permit the student to progress to writing simple programs on his own and be able to understand the specifications and instructions that accompany various factory-assembled prototyping boards.

The course presentation usually assumes some knowledge of digital electronics, but it skips over many of the fundamental concepts and theories so that the student can attain overall comprehension in the shortest possible time. The student is left to fill in fundamental knowledge, or study advanced texts as befits his individual needs.

These "crash" courses are not substitutes for more formal learning although they are pointing the way toward revision and rearrangement of the order in which the subject matter is presented in formal technical courses.

Some educators see the validity of introducing microcomputer training into programming and data-processing courses so that persons specializing in the field will have a better comprehension of the role of hardware, a subject now treated rather superficially in those specialized courses.

The public is anxious to learn to use computers, and microprocessors offer an expedient means of doing just that. A few manufacturers, like E & L, are responding to the demand, as are a few universities, with "hands-on" seminars, and the impact of their success will have far-reaching effects on how technical education will be taught in a technological future. ■

Introducing computers into schools requires more than just the availability of computers and a language for their use.

Interactive Computing in Secondary Schools in France

Robert S. McLean
The Ontario Institute for Studies in Education
Toronto, Canada

Perhaps one of the most interesting projects for the introduction of computers into secondary school curriculum is currently happening in France. It is interesting, both for the scale of the project and for its philosophical basis. The project is nationwide, involving the National Ministry of education, the National Educational Research Institute, five university centers, two computer manufacturers, and a large number of secondary schools. It is based on the belief that computer technology can make a contribution in all disciplines found in the secondary schools.

France has a very centralized educational system, rather unlike the organization found in North America. It has been said, not without justification, that the Minister of Education could look at his watch and thus know what was happening in any classroom in the country. That is less true today than previously, but there is still a large centralized influence over the content and activities of the individual classroom. Only in the 1973-74 school year did the Ministry institute the "10%" rubric under which individual lycees (secondary schools) were given ten percent of the school time to schedule as they wished. Accounts published in the newspapers indicated that many teachers had trouble filling this discretionary time with activities.

Within such a framework, the North American model of computer introduction by local initiative either at the school level or at the board of education level would seem impossible. Any substantial change in the curriculum and, particularly, in the facilities of schools must come from the top. The centralization and standardization also have great implications for the scale on which any innovation would have to be introduced.

The introduction of informatics (from the French "informatique" usually translated as "computer science", but used with a broader connotation here) began in 1970 with a joint initiative between the Ministries of Education and Industry. Together they formulated a plan for the introduction of computers into lycees, the training of teachers, and a coordination system for the development of curriculum materials that would be useful in many disciplines.

HARDWARE AND SOFTWARE

Introducing computers into lycees implied that suitable computing systems had to be available that could be installed in the individual school. Since the aim of the project was to create a system that could be used in many ways in addition to teaching about computers, an interactive system was specified, and a language for use with it was invented. The standard hardware system consists of a 16-bit minicomputer with 16K words of core, fixed head disk, one teletype with paper tape reader and punch, and eight CRT terminals. Two manufacturers produce the system: Companie Internationale pour l'Informatique (MITRA 15) and Telemechanique (T-1600). The systems sell for under $70,000 including the eight terminals.

These systems operate as single-language timesharing systems. The language, LSE (Language Symbolique d'Enseignement), was created especially for this application. While it has its roots in Algol, which is widely used in European universities and industry, it also has some of the appearances of BASIC. Programs consist of numbered lines, with editing and insertion of lines accomplished by the line number order rule as in BASIC; similarly, the line numbers serve as labels for transfer of control through explicit goto's.

The rest of the language is ALGOL-like in many respects except for the use of a loop scope rule that is FORTRAN-like, while retaining ALGOL conventions for the control of the incremented variable. Thus the loop command becomes "DO 27 FOR V ← v STEP p until f" or "DO 27 for V← v STEP p WHILE eb", meaning "execute all statements to and including number 27 for values of V starting at v and increasing by p, until the value of V passes f" in the first case or ". . . until eb is no longer true" in the second case. From ALGOL comes the conditional IF . . . THEN . . . ELSE . . ., along with the ability to make compound statements by enclosing statements between BEGIN – END pairs. Unfortunately, an entire conditional and its statements must appear on one line, thus not permitting very big compound statements.

Perhaps the most interesting feature is the fact that LSE also provides string manipulation and storage. There are three data types: real numbers, arrays of reals (up to 2 dimensions), and character strings. An operator is provided to concateinate strings, and 10 built-in functions are available for manipulating string variables and constants. These include functions to find the length of a string, extract a substring or group of letters, find the position of a substring in a string, ignore certain characters, convert numeric values to strings and vice versa, and to convert characters to and from their numeric codes.

The timesharing system provides the ability to execute program statements simply by typing them in, giving a powerful "desk calculator" facility and an interesting way to explore the language as well. This mode can also be combined with the running of a stored program, allowing a teaching program to pause while the student uses the terminal to perform some calculations, for example. The system allows the user to store and retrieve programs from disk, and to start execution (or restart after a pause) of a program. It is also possible to execute the program in trace mode, one line at a time, automatically printing the line executed on the terminal and pausing after each line. This is useful in debugging and in demonstrating the logic of a program.

Any terminal can obtain a copy of the current program being run at any other terminal at that moment.

The system allows any console to obtain a copy of the current program being run at any other console, in its state of execution at that movement. Both copies then exist independently and can continue to run or be examined, edited, listed, etc. This is useful when one wants to use the paper tape punch or reader on the teletype to read in a program, punch it out, or get hard copy. It is also useful in certain types of debugging and in setting up a class activity (the teacher can get the program ready and then have each student obtain a copy directly in its operational form).

A typical "computer room" is a converted classroom with the computer in one corner in a cabinet about 2' square by 6' high. The teletype used for listings and paper tape input/output is beside the computer. The desks have been arranged to support the terminals in two rows of 4 each. Since classes usually have 2 to 3 times as many students as there are terminals available, the use of programs by groups of 2 or 3 students at once is frequent, particularly for simulations. This arrangement appears to work well, stimulating human-human as well as human-machine interaction.

TEACHING THE TEACHERS

Introducing computers into schools requires more than just the availability of computers and a language for their use; it requires that teachers be trained in the use of computers and that curricular materials be developed that make use of the new technology in the various disciplines of the school. The intention of the Ministry in introducing computers was not to create another field of study — eg: computer science — but to integrate informatics with other studies as a tool.

Starting with very little computer expertise in the lycees in 1971, the Ministry has come a long way in developing the personnel capable of using the new technique in their teaching. The largest program has been a correspondence course taken by about 1600 teachers each of the three years it has been offered. The materials are in 12 chapters with 4 sets of homework sent in for marking. These teachers are released from their teaching duties for a three-day expense-paid visit to a regional computing center for intensive practical experience and lectures. (Hebenstreit, 1972) This has provided a large number of teachers who have a general knowledge of some of the basics of computing and its use in schools. The emphasis seems to be more on the computing and less on the applications.

The need for teachers with a higher level of expertise in computing is met by a system of "stages" (a "stage" — rhymes with "mirage" — is a period of training or retraining, generally undertaken by adults for a short period of time to learn a specific skill). Here a small group of teachers are sent to a university center for an academic year to participate in a special program that combines much of the traditional computer science course material with an emphasis on the problem of computer use in lycee curriculum. The extent of emphasis on these two concerns seems to vary among the five centers, and each seems to have a style of its own for accomplishing this instruction, running from very directive to very laissez faire.

The stages are open to teachers of all disciplines and the teachers are generally selected in a manner to encourage a range of disciplines to be represented at each center. During the 1973-74 academic year, the five programs included 68 "stagiaires" from mathematics, 39 from languages, letters and philosophy, 38 from physics, 31 from social sciences, 11 from natural sciences, 5 from industrial disciplines and 3 from artistic disciplines. The result is stimulating, but also presents a wide range of background that can challenge the instructors!

The trained teachers now form a resource for further strengthening the project. First, they are often in lycees that have been equipped with computers (there were 15 lycees equipped in 1974) and thus assume the management of this resource and the sensitization of their colleagues to the potential of the computer in their school. They do this partially on "release time" (20 to 24 hours a week per school) and in connection with their normal teaching duties. The Ministry estimates that the average computer is used about 30 hours a week or about 240 console hours per week. It is certainly not unusual to see all eight consoles occupied, often by pairs or trios of students.

Teachers often form discipline groups which work together to create packages of program materials.

Second, these teachers are in the best position to create curriculum materials. Most start fairly large projects during their stage and continue after returning to their schools; such a project is one expected result of the stage, and most can obtain additional release time the first year to complete the project. These teachers often form discipline groups which work together to create more substantial packages of program materials, especially where there is a high concentration of people in similar disciplines, as is the case around Paris. Groups in physics, natural science, and social science have been particularly active in the Paris area and have produced several noteworthypackages of material. Where the ex-stagiares are more separated, interdisciplinary groups meet occasionally to compare activities and this is often coordinated by the centers where the stages are held.

Program products and curriculum packages are distributed by the INRDP, The National Educational Research Institute in Paris, through two forms. The "Fiches Pedagogiques" (pedagogicalpapers) are packets of information about programs which have been developed for use in a particular curriculum. They contain descriptive data of the program, its use, and some evaluation of its success. Often the Fiches describe a set of programs that are logically related. Fifty Fiches had been published by the middle of 1974. These are distributed to all stagiares (old and new) and to all lycees equipped with computers. In addition, the programs to which they refer are available to the lycees in paper tape form.

A second publication of the INRDP is the Bulletin which is published bimonthly and contains reports of the working groups and of various meetings. It also has articles that have been contributed about uses of computers.

The materials created by these teachers depends largely on the ability of the individual teacher or of the disciplinary group to conceive of a use of computers or informatics in a curricular area and then to carry the idea through to completion as a curriculum package. In some fields, there is some advisory activity from post-secondary sources, particularly in the sciences.

Lafond (1974) lists the major work done to date as evidenced by the Fiches. In the field of letters and languages, fiches have appeared for the pedagogical use of an index, a study of the use of informatics concepts in the teaching of grammar, a program for the construction of

The computer is viewed as an adjunct to the curriculum and not a replacement for it.

sentences, and a program for conjugation and declension of Latin. In the social sciences, the study of demographic problems, automatic map making, and an introduction to economics through games of economic equilibrium and growth have appeared. In physics, programs for simulating experiments in optics (reflection), dynamics (inclined plane and gravitation) and thermodynamics (gas laws) have been distributed along with programs for the use of the computer to aid real experimentation. In natural science, one finds simulations of Mendel's law, genetic linkage, population growth, preditor-prey relationships, pollution problems, nutrition problems, as well as experimental aides. In mathematics, the materials include calculation programs, studies of the concepts of reflexivity, transitivity, etc., vector spaces, sets, limits, and probabilistic models. Interdisciplinary studies have offered games of inquiry.

As one can surmise from these products and from the size of the equipment available, the experiment is concentrated on the use of the computer as an adjunct to the curriculum and not a replacement for it. Thus, there is little interest, officially at least, in computer-assisted instruction as we know it. At the other end of the curriculum, computer science exploration by students is encouraged in informatics clubs which allow students with keen interest a chance to learn about programming. The primary role of the computer during school time is to execute programs devised by teachers to support their normal teaching activities. Relatively little programming by students appears to occur in the usual use of the computer.

THE FUTURE

The installation of computers in lycees is continuing, although not as quickly as was hoped originally, nor as fast as the teachers who have been on a stage would like. This is, of course, a phenomenon not limited just to France.

More notable, perhaps, is the evaluation program begun during the 1974-75 school year. Research by the INRDP includes a broad assessment not only of the curriculum materials created but also of the sociology of the introduction of computers in the lycees. The scale of the experiment itself and the scope of the evaluation planned for the experiment should yield some interesting conclusions about the extent to which secondary school curricula can be aided by computers as a classroom tool. ■

REFERENCES

Hebenstreit, J. Informatics in elementary and secondary education. In Bussell, B. *The Rio symposium on computer education for developing countries.* Rio de Janeiro: Ao Livro Technico, 1972.

Lafond, C. Rapport sur l'experience: "Introduction de l'informatique dans l'enseignement secondaire". Paris: INRDP, April, 1974 (Mimeo).

A MICROCOMPUTER SOFTWARE COURSE

by

Joseph C. Williams,
David S. Yaney
and
Robert K. MacCrone

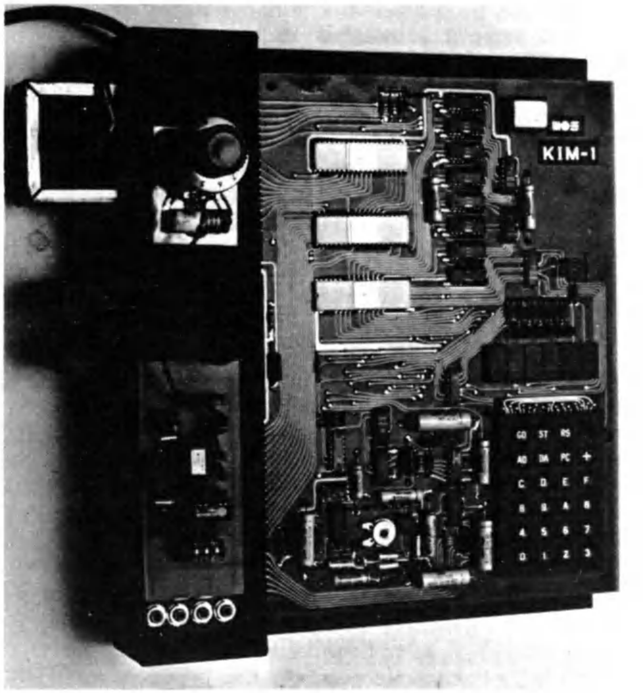
Materials Engineering Division
Rensselaer Polytechnic Institute
Troy, New York 12181

It's in control of a taxi meter that keeps track of the fares in a cab. It gives speech to an otherwise silent pocket calculator. It regulates and optimizes operation of some 1977 model cars. It has almost unlimited applications in business, industry and "smart" consumer goods. Best of all, it's cheap, reasonably fast, and easy to get.

The device is a microprocessor; a small computer on a chip. Combined with external memory and a chip to get data in and out, one can build a microcomputer of extremely small size and weight with surprising versatility. The problem is learning how to program the microprocessor. Its software requires the sort of programmer's art that has all but disappeared since the advent of larger systems and high level language.

Microprocessor software engineering was the subject of a 3-day short course held by the authors in Albany, New York, this past August. Twenty-five engineers, scientists, and students from East Coast states participated in writing and running programs on individual KIM-1 microcomputer systems. The KIM-1 from MOS Technology, Inc. is a complete, single board microcomputer with keyboard and display. Each participant took with him from the course the KIM-1, an Input/Output interface and power supply, and a working knowledge of their operation.

The participants had a wide range of applications in mind for microcomputers: a group of engineers from a machine



"Do they call this the hardware because its so hard to run?"

tool company wanted to design an engraving machine run by a microcomputer; a biologist needed a device to compare electrical signals from an animal's nervous system with previous signals. An electrical engineering professor wanted to introduce microcomputers to his students, and several participants planned to install microcomputers in their homes to control heating and lighting systems. Each person had a unique and complicated problem to solve, and although most of them had years of professional experience in their fields, using a microcomputer was a new task.

Using a microcomputer is in ways similar to using a larger system with the primary difference that the user must intimately know both the microcomputer being used and all devices to which the microcomputer is attached. Almost all programs for microcomputers are written in assembly language rather than a high-level language (like BASIC, FOCAL, or FORTRAN). While the general idea of writing a program from a flow chart is the same, the programmer must work with data byte by byte and often bit by bit. They may sound like a waste of the programmer's time and it would be if the problem to be solved involved, for example, complicated numerical calculations. (Most applications of microcomputers today do not require much "number crunching" for this reason.) Future microcomputers will have more extensive numerical abilities than those of today.

The starting point for learning to program microcomputers is the binary number system, which all present computers use, and the hexadecimal representation method for binary numbers. Once a student has gotten the idea that "hex" notation is just a convenient method of representing a pattern of binary "ones" and "zeros" he/she is in a position to begin learning how a computer works. To the programmer, the microcomputer appears as a group of registers, some memory locations with addresses, and an instruction set; the assembly language program specifies what operation is to be executed and on what register or memory location. Different microcomputers have different registers and instruction sets, so the user must program specifically for the machine being used. Although the program flow chart for the solution of a problem will not usually depend on the computer used, the assembly language program will. (A program written in BASIC to add two numbers will run on an IBM 370 or a PDP 8 while an assembly language program to do the same job will run on one or the other, but not both!)

The students in our microprocessor software course had all programmed computers using high level languages (mostly FORTRAN), but most of them had had no assembly-level experience. Two weeks before the course we sent them the first two chapters of notes so they could review the introductory materials. When the actual three day course began, we were writing and running programs almost from the start.

In addition to developing assembly language skills, a programmer must learn the procedures for Input/Output, interrupt, and timing operations with the microcomputer. Again, the details of these operations may differ from computer to computer, but the basic techniques are universal.

Once the programmer has knowledge of basic software techniques plus experience writing and debugging programs, actual applications may be considered. To become an outstanding microprocessor user one must keep the overall goals of the system in mind while dealing with the "petty" details of the software and hardware which can make or break the system. At the end of the three day course, the students had written and run programs using every feature of the KIM-1 microcomputer, and were ready to begin the design of systems incorporating microcomputers. The engineering of these systems must be with software as well as with the traditional hardware.

The authors were very pleased by the enthusiastic approach of the participants to the material and their rapid progress during the three days of intensive work. We are looking forward to the next running of the course in Albany, New York, on January 12th-14th, 1977.■

"Do you think I could get your program to self destruct?"

> **"To be practical, an education should prepare a man for work that doesn't yet exist, and whose nature cannot even be imagined."**
> **Charles E. Silberman**

The amount of computer power available at CMU can only be described as awesome.

Computer Science at Carnegie-Mellon Univ.

by Susan Hastings

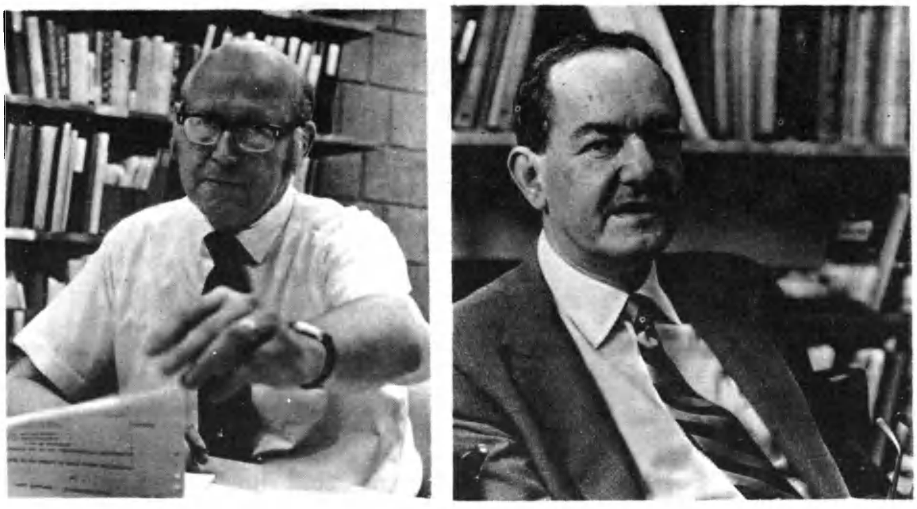
Carnegie-Mellon's Computer Science Department, ranked as one of the top three university departments in the country studying the computer, celebrated a milestone last October—its tenth anniversary. In 1965 the Department broke off from the university's Computation Center (which now acts as a processing service) in order to provide a home for those wishing to study the machine itself and to develop increasingly sophisticated hardware and software packages.

Then, as now, the Department had an interdisciplinary flavor, reflecting the research interests of the founding faculty. Through joint appointments with other academic units, department researchers have explored computer applications in architectural drafting, human speech understanding, and cognitive psychology, as well as other areas.

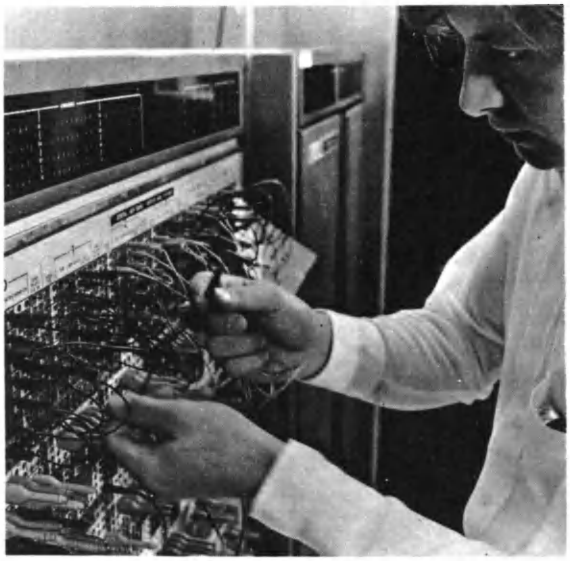
Paralleling the explosion of uses for the computer has been an intensive effort in the design and manufacture of hardware and software systems. One of the first interactive computers, the PDP 10, which allows a number of simultaneous users to communicate with it through keyboard terminals rather than punch cards, was developed by Professor of Computer Sciences and Electrical Engineering, C. Gordon Bell. Also developed at CMU were elements of the PDP-11, a mini-computer, and a remote terminal.

A speech understanding program currently underway at CMU has resulted in Hearsay I, the first functioning software system that enables a computer to understand spoken English and respond in it. Hearsay I was first demonstrated in 1972 as a chess playing computer. Electronic graphs depicted the acoustic properties of the human player's command and showed the system's progress in deciphering it, then the computer "spoke" its own move while it flashed the move on a TV set. Hearsay I has led to new insights about how the human mind deciphers speech and analyzes its meaning through context and definitions, acoustic sounds, and semantic and syntactic rules. Someday, as an outgrowth of this work and alternative programs in computerized speech understanding, even a child will be able to use a computer just by talking into it.

A more recent area of interest is the Image Understanding Project, a visual counterpoint to Hearsay's aural progress. Although visual information can now be encoded



Allen Newell (L) and Herbert A. Simon (R) from CMU jointly received the 1975 A.M. Turing Award, ACM's most prestigious award. Newell and Simon have made basic contributions to artificial intelligence, the psychology of human cognition, and list processing.



Shown here are four of the 16 PDP-11s tied together by CMU's multi-mini-processor system C.mmp.

electronically, e.g. TV sets or copying machines, the programming necessary to make a machine recognize and analyze this data, rather than simply store, transmit, or duplicate it, promises to be as complex as that required for speech understanding. One possible use of a machine that could interpret visual images at high speed is the analysis of weather satellite photographs, which currently can only be studied by humans.

These attempts to program computers with the equivalent of human perceptions have shown that a multiplicity of data bases, and the rules to use them, must be scanned and analyzed simultaneously to be able to provide useful information rapidly. For example, the computations required for speech understanding are so complex that it currently takes a computer at least ten to a hundred times longer than a human to understand what's been said. C.mmp is a multi-mini-processor developed at CMU which can simultaneously work on several aspects of the same problem. This new machine incorporates sixteen small PDP-11 computers and the software systems that will allow each of the different processing units to function independently, in tandem, or in a variety of flexible arrangements. C.mmp's parallel processing capability requires a different approach to programming, so Dr. William Wulf and his students laid down the fundamentals of a flexible operating system called Hydra, composed of processing programs and an executive program which controls the location, storage, and retrieval of data, and schedules jobs for the processors. They also developed a special compiler, BLISS, which translates instructions into machine language (binary code) and is used to implement Hydra.

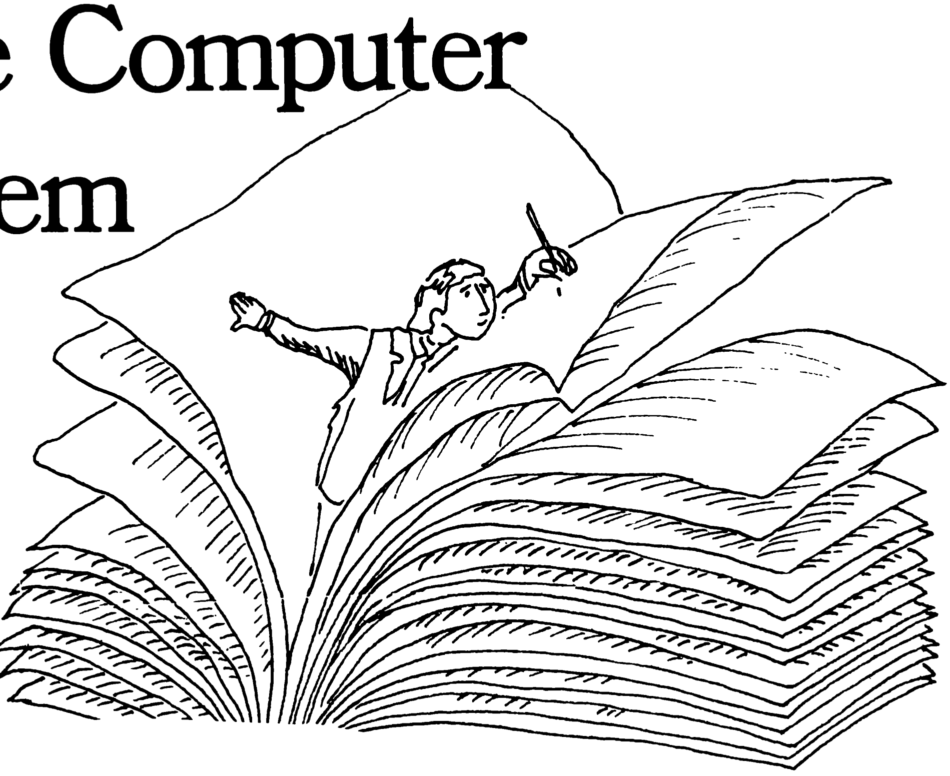
Still another research project at the Computer Science Department has resulted in the development of a combined hardware-software system which can produce a very detailed line drawing on a cathode ray tube that helps users to see the machine's progress as it seeks out a solution to a problem. Its most dramatic use is the production of strikingly clear pictures that seemingly move on their own. The Architecture Department is using this system in conjunction with the Computer Science Department in order to develop programs in which the computer will draw detailed architectural drawings automatically.

Most of the research programs described above depend upon the notion of artificial intelligence. A machine exhibits artificial intelligence when it reaches a decision similar to that of a human, even if it goes about reaching that decision in a different way. CMU's Computer Science Department has shown itself to be a leader in developing new uses for the computer through a great deal of cumulative human intelligence and hard work. Appropriately, its tenth anniversary was marked with a working technical scientific symposium, and not just a festive celebration.■

Photos: courtesy of Carnegie-Mellon University.

If you don't like the hardware available, you can always design your own.

FINAL EXAMS —Let the Computer Write Them



Bernard Eisenberg*

At one of its meetings earlier this year, the CUNY Board of Higher Education decided that units of The City University would move to a trimester system by September 1977. Kingsborough Community College formulated a modified trimester plan to start during the Fall 1976 semester. This plan was approved by the governing body at the College and the Board. The plan calls for an academic year consisting of two 12-week semesters, one in the fall and the other in the spring, and two six-week semesters, one in the winter and the other in the summer. We usually refer to this as the 12/6/12/6 plan. Without going into further details about the plan, it means that for each academic year, at least four final examinations will have to be prepared for each of the basic courses offered in all semesters. In addition, makeup finals will also have to be produced for those students who for legitimate reasons did not take the final examinations at the scheduled time.

The Old Way.

In the Math Department, of which I am a member, the usual procedure for preparing a final examination in courses having a large number of sections, is to form a committee of five faculty members who are teaching the same course. After three or four meetings at which the committee members and others teaching the course submit possible questions and comments regarding the examination, a draft of the final is assembled and circulated to all faculty teaching the course. The draft is then modified one or two meetings later after additional comments are made, and this becomes the final examination for the course. The examination is given to the typing service and after it is typed, it's reviewed by a faculty member for typing and mathematical errors. After these errors are corrected, the examination is photo-offset and enough copies printed to accommodate all sections.

In the evening session, each of the three or four faculty teaching these courses prepares his own final and has it typed and reproduced.

This process usually starts about halfway through the semester and deadlines are given for submission of the examination to permit typing and photo-offset. If the examination is not ready for typing on time, it becomes the responsibility of the faculty members involved to type, if necessary, and reproduce in whatever manner possible, the required number of final examination copies. Frequent notices from Deans are sent to the Chairmen and faculty reminding them about the deadlines. It is not unusual to have some final examinations submitted on the last day of the semester or even on the day of the final. This is one reason some Deans and Chairmen become bald rather fast.

Computer to the Rescue.

How nice it would be to have a computer produce a final examination in about three minutes by typing a couple of words into a teletypewriter connected interactively to the computer. Well, the day of the computerized math final is here and there appears to be no reason why this couldn't be done for any course in any discipline.

The Math 05 course entitled, "Intensive Math Review," was the first course selected for the preparation of a computerized final examination. This is a remedial course required of all students who don't do well on the placement examination. There are approximately 600 to 800 students enrolled in about 30 day and evening sections of this course during each of the regular semesters and about 100 to 150 students enrolled in four to six sections during the summer session.

---

*Dept. of Mathematics and Computer Science, Kingsborough Community College, City University of New York.

The course covers a review of arithmetic and elementary algebra. It offers no credit and has been taught for four hours a week during the regular sessions.

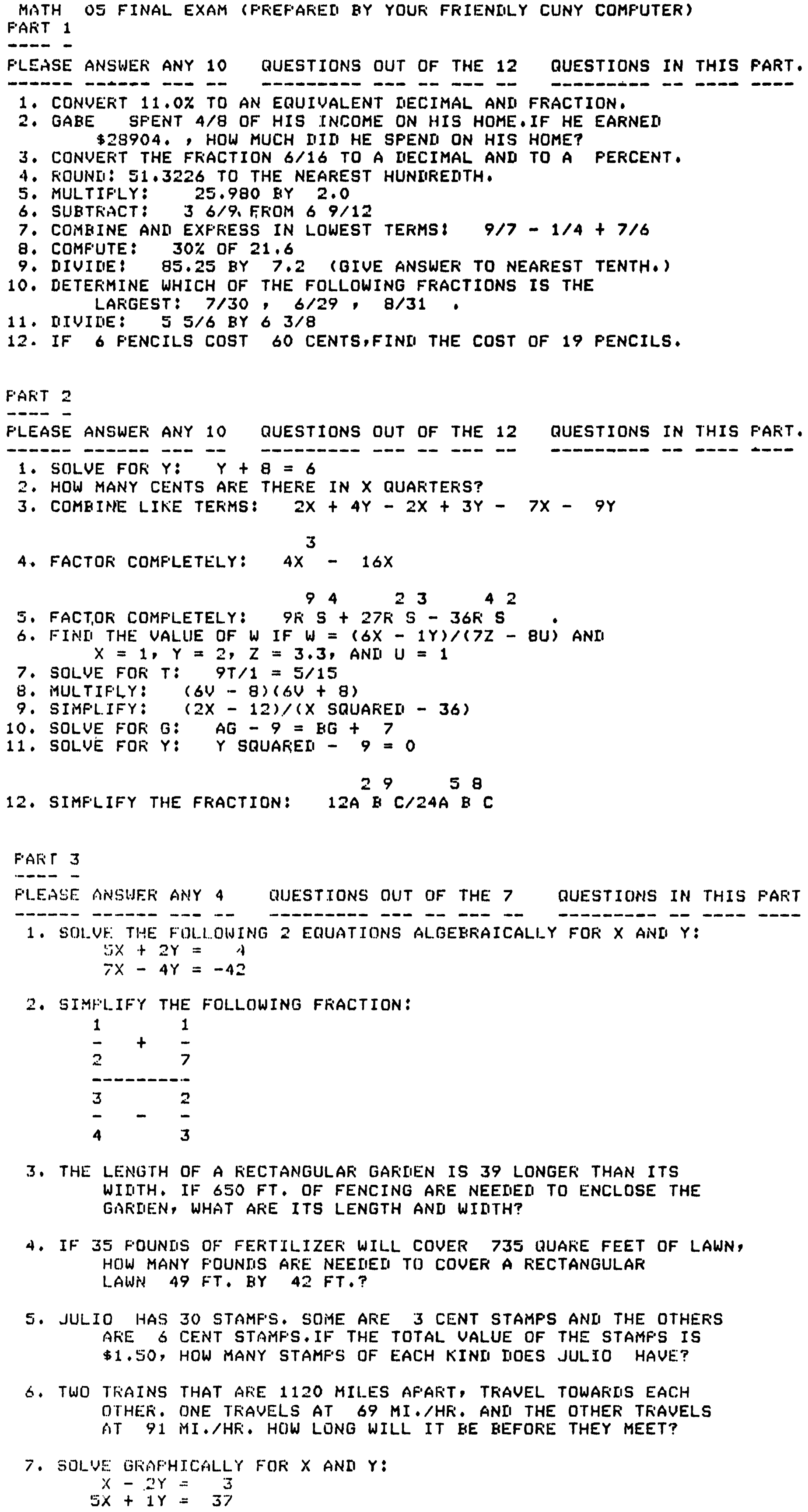
Specifications.

The specifications that had to be met by the computerized final were essentially those that should be met for any final examination.

1. The questions produced had to be diverse enough each time the computer was run so that they were representative of that term's work and the curriculum covered.
2. Although some questions might have the same wording from one examination to another, the numbers in the questions had to be different for each examination.
3. The numbers used in each of the problems had to be randomly suitable. This means that the numbers selected by the computer had to be such that solutions of the problems presented would not come out to six decimal places when integers were desired, that division by zero wouldn't occur in the problems and that unusually large numbers wouldn't appear when smaller ones were desired, and vice-versa. The numbers in the problems had to reflect those used in similar classroom exercises.
4. Each time a final is to be produced, the order in which questions appear must be random; that is, it should be equally likely for any question to be first or second or third, etc.
5. Provision should be made to give students a choice from those questions selected by the computer so that any deviations in covering the course material by the large number of faculty teaching it, would still permit the student to answer a sufficient number of questions from the curriculum covered by his/her teacher.
6. The program has to be flexible enough to incorporate new material that might be introduced from term to term or to delete questions no longer relevant.

In order to meet these requirements, the last six final examinations given in Math 05 were reviewed. There appeared to be a core of common questions in both arithmetic and elementary algebra that were repeated from term to term. For example, students were asked how to add, subtract, multiply and divide whole numbers, decimals, fractions and mixed numbers. Some questions required them to convert decimals to fractions and percents and vice-versa. They had to know how to apply these conversions in real-world problems. They were required to solve linear numerical and literal equations as well as quadratic equations. They were also required to know how to factor an

Here's one final exam prepared by the computer...



```
 MATH  05 FINAL EXAM (PREPARED BY YOUR FRIENDLY CUNY COMPUTER)
PART 1
---- -
PLEASE ANSWER ANY 10   QUESTIONS OUT OF THE 12   QUESTIONS IN THIS PART.
------ ------ --- --   --------- --- -- --- --   --------- -- ---- ----
 1. CONVERT 11.0% TO AN EQUIVALENT DECIMAL AND FRACTION.
 2. GABE   SPENT 4/8 OF HIS INCOME ON HIS HOME.IF HE EARNED
       $28904. , HOW MUCH DID HE SPEND ON HIS HOME?
 3. CONVERT THE FRACTION 6/16 TO A DECIMAL AND TO A  PERCENT.
 4. ROUND: 51.3226 TO THE NEAREST HUNDREDTH.
 5. MULTIPLY:    25.980 BY  2.0
 6. SUBTRACT:   3 6/9 FROM 6 9/12
 7. COMBINE AND EXPRESS IN LOWEST TERMS:   9/7 - 1/4 + 7/6
 8. COMPUTE:   30% OF 21.6
 9. DIVIDE:   85.25 BY  7.2  (GIVE ANSWER TO NEAREST TENTH.)
10. DETERMINE WHICH OF THE FOLLOWING FRACTIONS IS THE
       LARGEST:  7/30 ,  6/29 ,  8/31  .
11. DIVIDE:   5 5/6 BY 6 3/8
12. IF  6 PENCILS COST  60 CENTS,FIND THE COST OF 19 PENCILS.


PART 2
---- -
PLEASE ANSWER ANY 10   QUESTIONS OUT OF THE 12   QUESTIONS IN THIS PART.
------ ------ --- --   --------- --- -- --- --   --------- -- ---- ----
 1. SOLVE FOR Y:   Y + 8 = 6
 2. HOW MANY CENTS ARE THERE IN X QUARTERS?
 3. COMBINE LIKE TERMS:   2X + 4Y - 2X + 3Y -  7X -  9Y

                            3
 4. FACTOR COMPLETELY:   4X  -  16X

                            9 4     2 3     4 2
 5. FACTOR COMPLETELY:   9R S + 27R S - 36R S    .
 6. FIND THE VALUE OF W IF W = (6X - 1Y)/(7Z - 8U) AND
       X = 1, Y = 2, Z = 3.3, AND U = 1
 7. SOLVE FOR T:   9T/1 = 5/15
 8. MULTIPLY:   (6V - 8)(6V + 8)
 9. SIMPLIFY:   (2X - 12)/(X SQUARED - 36)
10. SOLVE FOR G:   AG - 9 = BG +  7
11. SOLVE FOR Y:   Y SQUARED -  9 = 0

                                 2 9     5 8
12. SIMPLIFY THE FRACTION:   12A B C/24A B C


 PART 3
 ---- -
 PLEASE ANSWER ANY 4    QUESTIONS OUT OF THE 7    QUESTIONS IN THIS PART
 ------ ------ --- --   --------- --- -- --- --   --------- -- ---- ----
  1. SOLVE THE FOLLOWING 2 EQUATIONS ALGEBRAICALLY FOR X AND Y:
        5X + 2Y =   4
        7X - 4Y = -42

  2. SIMPLIFY THE FOLLOWING FRACTION:
        1       1
        -   +   -
        2       7
        ---------
        3       2
        -   -   -
        4       3

  3. THE LENGTH OF A RECTANGULAR GARDEN IS 39 LONGER THAN ITS
        WIDTH. IF 650 FT. OF FENCING ARE NEEDED TO ENCLOSE THE
        GARDEN, WHAT ARE ITS LENGTH AND WIDTH?

  4. IF 35 POUNDS OF FERTILIZER WILL COVER  735 QUARE FEET OF LAWN,
        HOW MANY POUNDS ARE NEEDED TO COVER A RECTANGULAR
        LAWN  49 FT. BY  42 FT.?

  5. JULIO  HAS 30 STAMPS. SOME ARE  3 CENT STAMPS AND THE OTHERS
        ARE  6 CENT STAMPS.IF THE TOTAL VALUE OF THE STAMPS IS
        $1.50, HOW MANY STAMPS OF EACH KIND DOES JULIO  HAVE?

  6. TWO TRAINS THAT ARE 1120 MILES APART, TRAVEL TOWARDS EACH
        OTHER. ONE TRAVELS AT  69 MI./HR. AND THE OTHER TRAVELS
        AT  91 MI./HR. HOW LONG WILL IT BE BEFORE THEY MEET?

  7. SOLVE GRAPHICALLY FOR X AND Y:
        X - 2Y =   3
       5X + 1Y =  37
```

...and here's another generated by the same program.

```
 MATH  05 FINAL EXAM (PREPARED BY YOUR FRIENDLY CUNY COMPUTER)
PART 1
---- -
PLEASE ANSWER ANY 10   QUESTIONS OUT OF THE 12   QUESTIONS IN THIS PART.
------ ------ --- --   --------- --- -- --- --   --------- -- ---- ----
 1. IF 11 PENCILS COST  77 CENTS,FIND THE COST OF 24 PENCILS.
 2. SONNY  SPENT 4/9 OF HIS INCOME ON HIS HOME.IF HE E@RNED
        $29133. , HOW MUCH DID HE SPEND ON HIS HOME?
 3. DIVIDE:    4.46 BY  7.0  .(GIVE ANSWER TO NEAREST TENTH.)
 4. CONVERT 16.7% TO AN EQUIVALENT DECIMAL AND FRACTION.
 5. DIVIDE:   8 2/6 BY 8 7/8
 6. COMPUTE:   36% OF 32.4
 7. FIND THE AREA OF A TRIANGLE IF ONE SIDE IS  7 FT. LONG AND
        THE ALTITUDE UPON THAT SIDE IS 12 FT.
 8. ROUND: 67.7431 TO THE NEAREST HUNDREDTH.
 9. MULTIPLY:    25.790 BY  4.3
10. AN ARTICLE SELLS FOR  $20.12. A 6% SALES TAX IS ADDED.
        WHAT IS THE TOTAL PAID FOR THE ARTICLE?
11. COMBINE:   146.428 -  5.73 +9.7
12. SUBTRACT:   6 5/9 FROM 9 1/12


PART 2
---- -
PLEASE ANSWER ANY 10   QUESTIONS OUT OF THE 12   QUESTIONS IN THIS PART.
------ ------ --- --   --------- --- -- --- --   --------- -- ---- ----
 1. HOW MANY CENTS ARE THERE IN X QUARTERS?
 2. SOLVE FOR G:   AG - 6 = BG + 12     .
 3. FACTOR COMPLETELY:   B SQUARED - 2B -  8
 4. WRITE AN EQUATION FOR THE FOLLOWING WORD PROBLEM
        BUT DO NOT SOLVE.  9 TIMES A NUMBER IS 4 MORE
        THAN 4 TIMES THE NUMBER. FIND THE NUMBER.
 5. SOLVE FOR Y:   Y + 6 < 1
 6. SOLVE FOR Y:   Y SQUARED - 36 = 0

                                4 9     7 2
 7. SIMPLIFY THE FRACTION:   36A B C/24A B C   .
 8. COMBINE INTO A SINGLE FRACTION:   8/(9X) - 9/(45X)
 9. COMBINE LIKE TERMS:   1X + 5Y - 3X + 8Y -  5X - 12Y

                           8 7     2 3     4 2
10. FACTOR COMPLETELY:   3R S +  9R S - 12R S    .
11. MULTIPLY:   (3V - 7)(3V + 7)
12. FIND THE PERIMETER OF A TRIANGLE WHOSE SIDES ARE
        (3X + (-4Y)),(-2X + 8Y), AND ((-9Y) - 7X)


PART 3
---- -
PLEASE ANSWER ANY 4    QUESTIONS OUT OF THE 7    QUESTIONS IN THIS PART.
------ ------ --- --   --------- --- -- --- --   --------- -- ---- ----
 1. IF 54 POUNDS OF FERTILIZER WILL COVER 1782 SQUARE FEET OF LAWN,
        HOW MANY POUNDS ARE NEEDED TO COVER A RECTANGULAR
        LAWN  77 FT. BY  84 FT.?

 2. SIMPLIFY THE FOLLOWING FRACTION:
        1       1
        -   +   -
        4       9
        ---------
        3       2
        -   -   -
        4       3

 3. SOLVE THE FOLLOWING 2 EQUATIONS ALGEBRAICALLY FOR X AND Y:
        7X + 6Y =  14
        7X - 1Y = -35

 4. THE LENGTH OF A RECTANGULAR GARDEN IS 21 LONGER THAN ITS
        WIDTH. IF 350 FT. OF FENCING ARE NEEDED TO ENCLOSE THE
        GARDEN, WHAT ARE ITS LENGTH AND WIDTH?

 5. MARCEL HAS 50 STAMPS. SOME ARE  4 CENT STAMPS AND THE OTHERS
        ARE  7 CENT STAMPS.IF THE TOTAL VALUE OF THE STAMPS IS
        $3.20, HOW MANY STAMPS OF EACH KIND DOES MARCEL HAVE?

 6. A COLLECTION OF 54 COINS CONSISTS OF DIMES AND QUARTERS
        AND HAS A VALUE OF $10.80. HOW MANY OF EACH ARE THERE?

 7. ONE NUMBER IS 5 MORE THAN 7 TIMES A SECOND NUMBER.
        THEIR TOTAL IS 29. FIND THE NUMBERS.
```

algebraic expression, how to apply this in the solution of equations, and how to solve a system of two linear equations algebraically and graphically. They had to perform the arithmetic operations on algebraic fractions and expressions and, finally, they had to know how to solve a variety of word problems algebraically.

Program Development.

To develop a computer program to simulate these exams, it was decided to divide the examination into three parts. The first would cover arithmetic, the second would cover elementary algebra, and the third would contain word problems, simplification of a complex fraction and problems requiring the solution of two simultaneous equations algebraically and graphically. A bank of 18 arithmetic questions was prepared for Part 1. A bank of 18 elementary algebra questions was developed for Part 2 and a bank of 10 questions was selected for Part 3.

Here's how the program works. Let's assume that the faculty have decided that the students should answer 10 out of 13 questions in Part 1, 10 out of 13 in Part 2 and 4 out of 7 in Part 3. Let's further assume that such a decision by the faculty will not take more than a day. The latter assumption may be somewhat risky, but not unreasonable. On the first day of classes or during the first week, the program (developed in BASIC) is run using the aforementioned selectivity numbers and within three minutes, the final examination produced is the final examination that the student gets. This examination will contain randomly-selected questions from each bank, they will be randomly presented in the examination, and each program run will contain different and suitable numbers for each of the questions. (Two sample copies produced in a six-minute period are shown.) The computer printout of the examination is given to the reproduction department for photo-offset and as far as the faculty are concerned that final is finished. No further typing is needed.

Benefits.

Faculty don't have to spend long hours meeting and bothering each other to meet deadlines. The time saved by the faculty can be devoted to their students and teaching rather than to the chores of preparing the final. To improve this program still further, one could make it a multiple-choice-type examination which could be machine-graded. This however, would do the student a disservice, in math courses at least, since we're interested in the work done and method used to support the answers attained. The reasoning and thinking used to solve a problem are more important than the answers.

Time Saved Using Computerized Finals

A reasonable estimate of time saved in a semester by using computerized finals for Math 05 is as follows:

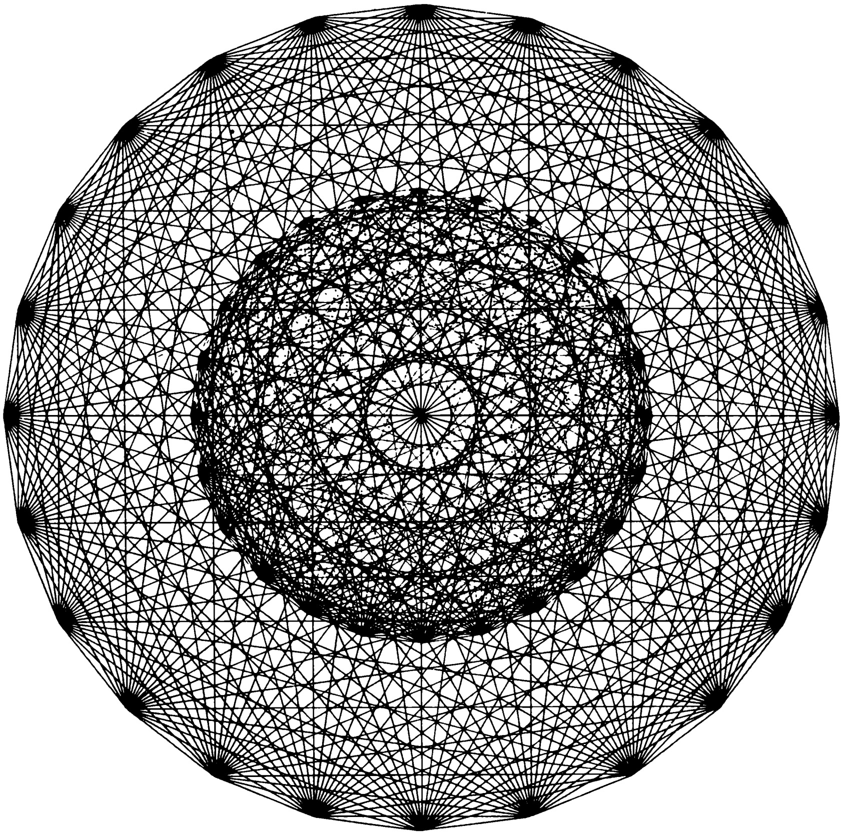
Faculty time for final exams: 50 hours/semester — day session
15 hours/semester — eve. session
Typing time for final: 2 hours — day session
6 hours — eve. session
Faculty time for 3 classroom exams: 72 (assuming 2/3 of faculty participate)

With savings of this magnitude, each regular semester and probably a third to half as much for each six-week winter and summer session, it would appear economically feasible to computerize all final and classroom exams for basic courses in each discipline. The larger the number of multiple sections for a course, the greater will be the faculty time saved.

Nevertheless, a second program has been produced which not only presents questions and suitable numbers randomly, but also gives the answers to each of the questions. This also saves some faculty time when they prepare to grade the examinations. For the benefit of bilingual students, a third program will produce a random final examination in Spanish, with answers as well.

Other Savings.

There are other savings and benefits from the use of computer-prepared random final examinations that will accrue to faculty and students as well. During the last two weeks of each semester it is customary for faculty to prepare a review sheet that serves as a guide for students in preparing for the final. The review sheet usually highlights those topics the student will be responsibile for and will most likely contain a list of problems he/she should know how to solve. The faculty members no longer have to do this. Once again, on the first day of the term, instead of during the last two weeks, the complete bank of questions in all three parts of the examination can be run off by the computer in about three minutes and again the questions are randomly positioned in each part and the numbers used are suitable and randomly selected by the computer. In fact, in those questions where a person's name is mentioned, the name is also randomly selected. The complete set of questions may be given to the students and they can be told that the major portion of the final examination will be selected from this set. The goals for the course are thereby set at the first session. As an alternative to giving the students the entire bank of questions, a separate bank of 30 mixed problems in arithmetic and algebra is available and can be used.



It frequently happens that with large groups of students taking the same course, the final exam is usually administered in large lecture halls with students occupying alternate seats. By using three different computerized finals for the same course, all the seats in the lecture halls may be used and final exam number 1 may be given to those students occupying columns 1, 4, 7, 10, etc., exam number 2 may be given to those in columns 2, 5, 8, 11, etc., and exam 3 may be given to those in columns 3, 6, 9, 12 etc. As a consequence, fewer proctors will be required to administer the exam and student cooperation on the exam is minimal.

Improving the Quality.

To improve the quality of the exam, the bank of questions in each part will be increased by at least 50 percent during the current semester so that there will be at least 27 questions in each of Parts 1 and 2 and at least 15 in Part 3. Any other suggestions from faculty and students who use these exams this semester will be incorporated in the next master program. Other future plans call for computerizing the three exams given during the semester. Since an exam is usually given after several topics are covered, the bank of questions from which each of the classroom exam questions is selected will be solely from these topics. Starting with the winter session, all classroom and final exams in Math 05 may be computerized with and without answers and bilingually in Spanish. All makeup exams in Math 05 which have to be prepared, can be done so in less than three minutes. Although not all faculty (about 13) teaching Math 05 may wish to join this project for one reason or another, the initial reaction appears to be that most, if not all, will. All of the evening-session final exams can also be run on the computer and a different one may be given to each evening session instructor.

Future projects will consist of the development of similarly computerized exams for such multiple-sectioned courses in Math and then in other disciplines. ■

Computational Unsolvability

Reprinted with permission from *Science News*, the weekly news magazine of science and the applications of science Copyright 1976 by Science Service, Inc

Problems where complexity grows exponentially are now believed incapable of exact solution on even the fastest possible computers

BY LYNN ARTHUR STEEN

Problems of arrangement and scheduling are among the most common yet most vexing in all of applied mathematics. Sometimes solutions are intuitively obvious; other times they are surprisingly paradoxical. Some problems succumb to prosaic methods of attack, while other apparently similar problems are totally intractable.

The common managerial task of scheduling a given set of jobs among available staff in order to finish the work in the least possible time is a good example. Ordinarily, if the manager sees that he can't meet his deadline with existing staff, he will add an additional person. But in some cases, this additional person might increase rather than decrease the total length of time required to finish the job. The subtleties of scheduling are so deep that, although it is possible to write a computer program that will determine the most efficient schedule for a staff of two, it appears that the same problem for a staff of three requires so much computer time as to be, for all practical purposes, impossible to execute when the number of jobs is large.

Recent research has revealed a profound dichotomy in the nature of these combinatorial problems. Some are, in a specific technical sense, easy, while others are hard. The latest major problem to be successfully diagnosed in these terms is over 100 years old. It begins in a tale of four cities and ends in current research in computer networks and integrated circuit design.

Suppose an engineer wants to find the least expensive way to join cities located at the four corners of a square with a network of telephone cables. Since the cost of installing the network is roughly proportional to the total length of cable, he might expect that the best solution is to lay cable along the diagonals of the square, with a junction in the middle. But, surprisingly, this does not yield the shortest (and cheapest) network: the engineer could do about 3 percent better if he used two junctions and joined the cables at angles of 120° (see diagram). This configuration is the optimal, or best possible, solution to the engineer's problem. Because this solution was first studied by the 19th-century German geometer Jacob Steiner, the junction points where three paths meet at equal angles are today called Steiner points. (The 120° requirement for minimal path lengths is the two-dimensional analogue of the 120° angles at which soap films meet [SN: 9/20/75, p. 186].)

Modern engineers confront varieties of Steiner phenomena in problems ranging from the design of microprocessor chips to nationwide communication networks: The determination of the shortest network linking certain given vertices is one of the famous unsolved problems of combinatorial mathematics known as the Steiner minimal tree problem. (It is called a tree problem because of the resemblance between its solution network and complex branching of trees.) Although the nature of the general solution is well known, finding locations for the required Steiner points is a very difficult problem.

Until 1961 it wasn't even known if the problem could, at least in principle, be solved by a search of all possibilities. At that time Z. A. Melzak invented an algorithm (a step-by-step solution procedure) that introduced possible Steiner points in a sufficiently systematic way that it would eventually find the optimal configuration. But, despite a variety of improvements since then, Melzak's algorithm takes so much time that even on the fastest computer it is really feasible only for networks with about 15 to 20 vertices.

An experienced designer could do just about as well "eye-balling" the problem, that is, examining a scale drawing and introducing Steiner points where it looks as if they will do the most good. What makes the Steiner minimal tree problem so difficult is trying to tell a computer which of the many possible Steiner points "look good." It is precisely when the problem gets too large for a person to "eye-ball" that the computer is most needed, and that is precisely where existing programs are of no use.

The meager results of nearly two decades of work on the Steiner minimal tree problem led many researchers to speculate that the problem was in fact intractable. That it is indeed intractable has now been proved by Michael Garey, Ronald Graham and David Johnson of Bell Laboratories in Murray Hill, N.J.

To understand the nature of their result we need to examine briefly the nature of computer algorithms that are used to solve problems of scheduling or arrangement. Solving such problems involves searching through different combinations of events in space or time; thus these problems are part of what is known as combinatorial analysis. All such problems have in common a natural "tree structure" in which early tentative decisions by the solver lead to branch points in the solution process where several other options may be pursued.

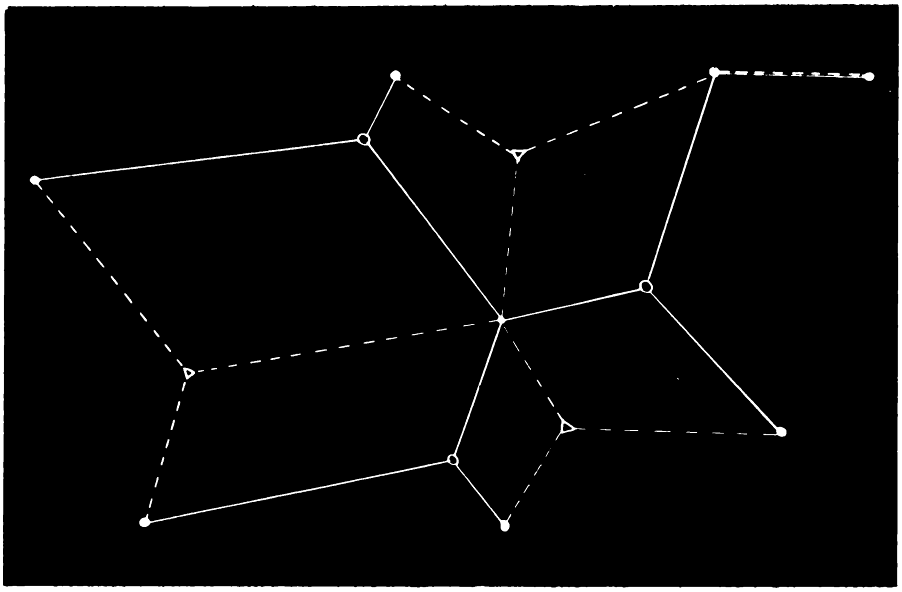
In a typical problem, there may be only a few correct routes through an enormous maze of branches. A combinatorial problem is like a huge tree with fruit at the tips of just a few twigs. How is a near-sighted bug crawling up the trunk going to select only the branches that lead to the fruit?

One way would be to have an oracle who can see the whole system at a glance. This is, in fact, how many relatively small combinatorial systems are solved—by a Gestalt-mathematician who apprehends the solution in a single act of perception. But it does not work for large systems because computers lack human insight, and humans lack the computer's memory.

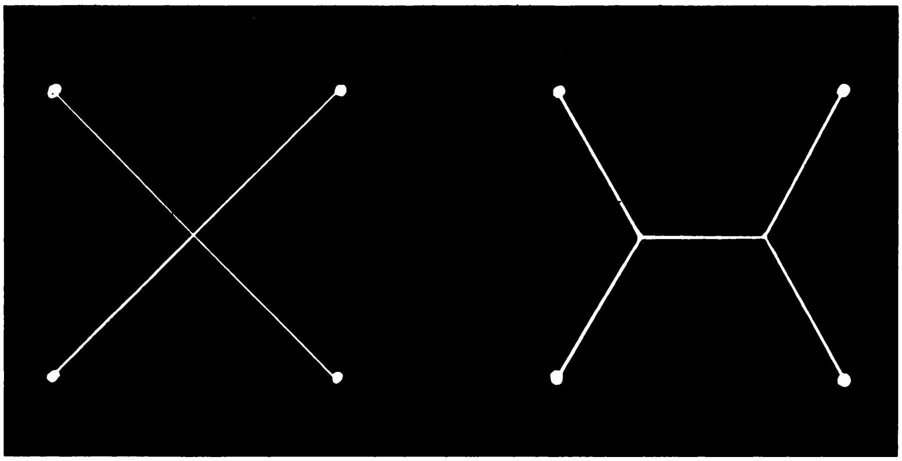
A second method would be to employ a so-called nondeterministic algorithm. "Nondeterministic" is used to describe methods that avoid the problem of determining the correct branch by following all branches simultaneously. This requires, hypothetically, a computer that replicates itself over and over again so that late in the process thousands of similar programs will be working alongside each other, simultaneously pursuing different branches of the solution tree. Whenever any one finds the fruit, the problem is solved.

Nondeterministic algorithms have a natural advantage of speed, for the time required for such an algorithm to solve a particular problem depends only on the total path length from beginning to end (called the depth of the solution tree) and not at all on the number of different branches in the tree. But they achieve this advantage by unrealistic simultaneous replication of computation power. Only in the last few years have parallel-processing computers been developed, and the number of parallel tracks is strictly limited by the nature of the hardware. So, for all practical purposes, nondeterministic algorithms are a figment of a theoretical imagination. They cannot be used for practical solution of large combinatorial problems.

The conventional way to simulate par-



Cities ● connected with a network by means of Steiner points ○ are pictured by the solid line. An alternative Steiner network linking the same cities is marked by dotted lines, using the Steiner points △. Both of these Steiner networks require less total distance than any straight-line network that did not use Steiner points, but it is not immediately clear which of these two Steiner networks uses least total distance.



Two ways to efficiently connect cities with telephone networks. For a square one mile on each side, the obvious method on the left requires $2\sqrt{2} = 2.83$ miles of cable. The more subtle means on the right, using cables that intersect with equal angles at two junctions, required only $1 + \sqrt{3} = 2.73$ miles of cable.

allel processing in a standard sequential computer is by means of a "backtrack" algorithm. Each time a branch point is reached, one branch is pursued and the others are stored in a stack of incompletely developed options. When the program reaches a dead end along the path it is pursuing, it backtracks to the most recently encountered unexplored branch stored in its options stack and pursues it. This process is repeated as often as necessary until one of the sequences of branches leads to a successful conclusion.

Backtrack programming (and related techniques called "branch and bound" algorithms) are just systematized approaches to trial and error: They organize the trials to ensure that errors, once discovered, are never repeated. But even a systematic search of all possibilities requires an enormous amount of time. Typically, the time required for a backtrack solution grows exponentially with the size of the problem.

The facts of exponential growth overwhelm even the astonishing speed of modern computers. Typical combinatorial problems involving, say, 10 cities (or 10 tasks to be scheduled) may require about 2^{10} (roughly, 1,000) branches. Depending on the complexity of the investigation to be carried out along each branch, this can be done in about a second or so of computer time. But if the problem increases to size 50—not all uncommon—the branching increases to 2^{50}; at 1,000 branches per second, this would take over 30,000 years! Thus computer scientists view as intractable large problems whose only known solutions are achieved by backtrack programming. They are computationally unsolvable.

Much better are those algorithms that grow polynomially rather than exponentially. If the time involved is of the order of n^2 or n^3 rather than 2^n, the computation time for large problems is dramatically reduced. For example, to continue with the hypothetical computations of the previous paragraph, a computer checking out 1,000 branches a second would manage 10^2 branches in a tenth of a second, and 50^2 branches in 2.5 seconds. Even 50^3 branches would only take two minutes. So to get a rough measure of the time required to solve a problem, computer scientists look first at whether the solution algorithm grows polynomially or exponentially with the size of the problem data.

Exponential growth most often results from a solution tree that is too broad: Even though the number of steps in a correct solution may grow polynomially, if the number of unfruitful branches that must be explored is too great, the time required by the solution algorithm may grow exponentially with the size of the problem. Problems like this can be solved in polynomial time by a nondeterministic algorithm. But, as we have seen, such algorithms are idealizations, incapable of actual implementation in polynomial time.

The class of problems that can, in principle, be solved by a nondeterministic algorithm of polynomial time—a class called NP, short for nondeterministic polynomial—thus includes many of the problems whose solutions actually seem to require exponential time. Whether in fact those problems that now seem to require exponential time actually cannot be done in polynomial time is not known: One of the major unsolved problems of current computer science is whether, possibly, every problem in the class NP can really be solved in polynomial time by an ordinary (deterministic) algorithm.

Massive circumstantial evidence has led virtually all informed observers to the conclusion that some problems in NP cannot be solved in polynomial time. This conclusion, if upheld, means that large problems of this type cannot be solved at all.

The first step in this chain of evidence was taken in 1971 by Stephen Cook of the University of Toronto who showed that each problem in the class NP can be transformed into a certain problem in mathematical logic, called the Satisfiability Problem, in such a way that any algorithm that would solve the Satisfiability Problem could be adapted to solve the other problem as well. (The Satisfiability Problem is the question of whether a Boolean logical expression can be satisfied [i.e., made true] by appropriate choice of the propositions from which the expression is built.) Cook's result shows that, in some sense, no problem in the class NP is any harder than the Satisfiability Problem.

Shortly after Cook announced his result, Richard Karp of the University of California at Berkeley showed that many other problems in the class NP share the distinction of the Satisfiability Problem. He called these problems NP-complete; they are sufficiently detailed to serve as prototypes for all other NP problems. Each NP problem can be transformed into any NP-complete problem and solved by appropriate adaptation of the solution algorithm for the NP-complete problem.

NP complete problems form a subclass of the class NP containing those of maximum difficulty. Karp (and others after him) showed that many famous problems of finite mathematics are in this class. These include the famous "traveling salesman problem" (find the shortest route that visits each city on a list exactly once), "0-1 integer programming" (linear programming in which variable values are limited to yes or no options) and "graph coloring" (assign a limited number of colors to regions in such a way that no regions with a common frontier receive the same color). The recent result of Garey, Graham and Johnson shows that the Steiner minimal tree problem is also of this type: It is NP-complete.

Most of the problems now known to be NP-complete have an extensive history of unsuccessful search for a polynomial-time algorithm. Recognition that they belong to the class of NP-complete problems shows that they are essentially equivalent problems. Thus the accumulated evidence of unsuccessful search for efficient algorithms for each of the several dozen NP-complete problems concatenates into an impressive record of failure.

Nearly half a century ago the mathematical logician Kurt Gödel astonished the mathematical and philosophical world by showing that in any sufficiently complex mathematical system there will always be intrinsically undecidable propositions—statements that can, by their very nature, never be proved or disproved. The status of NP-complete problems—if present beliefs are proved true—is somewhat analogous: They are problems that are sufficiently complex that, by their very nature, they cannot be solved in any practical amount of time. Gödel's work established the existence of problems that are theoretically unsolvable; NP-completeness points to the existence of problems that are computationally unsolvable.

Gödel's work on undecidable propositions led logicians away from a fruitless task (the complete formalization of all mathematics) and into more promising terrain. Similarly, the discovery of NP-completeness is right now turning applied combinatorial mathematics from the search for exact algorithms to the search for sufficiently good approximate ones. With this new focus comes a whole host of new and interesting questions concerning the establishment of standards by which an algorithm can be judged when we know that it is in the nature of things that it cannot be perfect. ■

State-of-the-Art vs. Compatibility

Although I have been a computer user since the IBM 650, Bendix G-15 and other assorted antique relics, I am not, by any stretch of the imagination, a Hardware Giant. Or Software Giant either. Indeed, all I want is a nice docile machine which will do my bidding and be my tool without me having to learn about its guts. In other words, I want to get in the car, turn a key, and drive off.... I don't want to build, repair, or even understand the durn thing.

But unfortunately it appears that the manufacturers in the microcomputer industry don't see it quite this way. First of all, most of the companies were started by engineers or programmers, exceptionally talented from a technical standpoint. Each one has a slightly better mousetrap, never mind that it's not quite compatible with one introduced three months before by another company or even their own firm. So we have a rapid succession of fantastic, state-of-the-art products, most of which don't quite work with the system you bought a couple of months ago. In fortunate cases only a few components are obsoleted; perhaps a marginal board can be updated. But remember, I'm just a user, so forget about bypassing the 7504 and putting in a 7807. Or rewiring my cassette recorder so the microphone switches off at full treble instead of full bass. Mumble, mumble. Unfortunately all too often the succession of state-of-the-art advances from the manufacturers looks like ERISA to the consumer. (ERISA to the Feds means Employee Retirement Income Security Act; to DP types it's a software package; however, as used here it means "Every Ridiculous Idea Since Adam.")

I could cite any number of real live examples revolving around the five *Creative Computing* systems but I really don't want to single out any five or six manufacturers — the problem is so very pervasive. For example, in tape cassette formats on our systems alone we have:

1. CUTS (Processor Technology SOL-20)
2. Altair (MITS. We don't worry much about this one; it hardly ever works, but we keep trying).
3. Xitan (TDL)
4. Byte/KC (SWTPC)
5. Tarbell

And why, he asked rhetorically, should software written for the Cromemco Z-80 CPU not work on the TDL Z-80 ZPU? Ha! Only the beady-eyed programmer knows. Or why, he asked cynically, should Seals and ECL 8K static memory behave entirely differently in a Dazzler-equipped system? Double ha!

Not that I should be giving advice to manufacturers and retailers, but in their position I think I would ask: are the expected sales as a result of introducing this latest state-of-the-art advance greater than the expected lost sales as a result of incompatability with existing systems?

In the short run, unfortunately, the burden is on the customer (and perhaps on the retail computer stores) to figure out if board X will work with system Y. Caveat emptor.

David H. Ahl

All about recursion and various other goodies offered by APL and even some versions of BASIC.

Something is Missing . . .

Craig A. Finseth

BASIC, as a programming language, offers a wide variety of features. There are, however, some features that most BASIC systems do not offer. Recursion and "on the fly" variable creation/deletion are two examples of these. In other languages, such as APL, these concepts are built-in as part of the language. Why is an APL system able to do these things while most BASIC systems cannot?

The prime reason for the difference is that APL stores its data in a way that is much more flexible, as well as making provisions for recursion. There is a price paid, in that the format for storing data can become rather complicated. This article is intended to make the concepts behind APL's data format visible. Don't expect to be presented with a "cut and dried" method for doing what would not be useful in itself. Instead, ideas and some "tools of the trade" will be evolved. In this way, you will be shown the reasons for a method of operation, as well as some other methods that do not work as well and the reasons why they don't.

It should be noted that this article is written about a specific application for these concepts. Because of this, there will be assertions made that something "must" be done a certain way. As with many things in the computer field, there are usually many other ways of doing something and some of them are probably better. Don't think that a certain way is the only way: think about *why* you are doing it that way and look for better methods.

Recursion

Recursion is the first difference mentioned, so let's start with it. First, what is recursion? A function is recursive when it calls itself. (It can also call a second function which calls the first, etc.) We shall use the classic factorial function example to illustrate the idea. (Note that this method is not necessarily the most efficient way of calculating factorials!)

Now, what are factorials? A factorial is the product of the integers from one to n. Thus, five factorial, written 5!, means 5*4*3*2*1. In the example we shall take advantage of a useful property of factorials in that n! is the same as n*(n-1)! (that is, 5! is the same as 5*4!). So, in our factorial function we can just have it call itself to figure out what (n-1)! is and then multiply that result by n. If you think that this could go on forever, you're right. Fortunately, mathematicians have conveniently defined zero factorial as one (0!=1). It's an easy matter to check to see if the argument is a zero; if so, we just return a one. One method of writing this function in a BASIC that allows multi-line function would be thus:

```
10 DEF FNF(N)
20 IF N<>0 THEN 50
30 LET FNF=1
40 GOTO 60
50 LET FNF=N*FNF(N-1)
60 FN END
```

This should look quite straightforward, with the exception of line 50 which is where the function calls itself. For this example, we are assuming that our BASIC system is able to handle recursive functions.

Suppose that we were to try to find 3!. Someplace in a program we might have a statement like the following:

```
666 LET X=FNF(3)
```

What would the execution of FNF look like? One way of showing what happens is to make a second copy of the function whenever it calls itself (this is also a possible method of implementing recursion). Figure 1 shows what it would look like.

In it, we inserted in each call to FNF a copy of the function. Along with this insertion, in each copy we replaced N by (N-1). (Remember that we want to do (n-1)!). Mentally setting N to 3, we can follow down the chain until we reach the end. (This diagram is tailored to the specific case of N=3. If N were to be ten, then there would be eleven copies of the function.) In the fourth copy, we see that (((3-1)-1)-1) is, indeed, zero and so we return a one. This one is multiplied oh ((3-1)-1) and the copy returns a one again. This result is multiplied by (3-1) which yields a two and that is finally multiplied by three, giving the final value of six. Three factorial is, of course, six.

Even though we can get the right answer this way, making copies of functions is not a very good way to operate. One reason is that you eat up both time and storage because you have to keep making copies of the function. Another reason is not to obvious. When we made copies of the function, we just substituted (N-1) for N whenever necessary. In our specific case, this worked fine, but how do we handle the case of the statement LET N=5? This would translate to LET (N-1)=5, which looks fishy, to say the least. If we were smart, we would change this to LET N=5+1, but this doesn't work on such other cases as LET(ABS(N))=5, not even to mention LET(A+B)=5. How do we translate that? Is N now equal to +5 or -5? It looks as if we need a new way of keeping track of what's happening, since it is not only inefficient, but nearly impossible to implement "blindly" as has been done here.

```
10 DEF FNF(N)
20 IF N <> 0 THEN 50
30 LET FNF=1
40 GOTO 60
50 LET FNF=N*   10 DEF FNF( (N-1) )
                20 IF (N-1) <> 0 THEN 50
                30 LET FNF=1
                40 GOTO 60
                50 LET FNF=(N-1)*  10 DEF FNF( ( (N-1)-1) )
                                   20 IF ( (N-1)-1) <> 0 THEN 50
                                   30 LET FNF=1
                                   40 GOTO 60
                                   50 LET FNF=( (N-1)-1)*  10 DEF FNF( ( ( (N-1)-1)-1) )
                                                           20 IF ( ( (N-1)-1)-1)  <>  0 THEN 50
                                                           30 LET FNF=1
                                                           50 GOTO 60
                                                           50 LET FNF=( ( (N-1)-1)-1)*FNF(...)
                                                           60 FN END
                                   60 FN END
                60 FN END
60 FN END
```

Fig. 1. Using recursion to calculate factorials.

Stacks.

Let's digress for a moment and talk about food. In many cafeterias, there are spring-loaded platters in the counter that support stacks of plates, with only the top one visible. Suppose that we write a three on one plate and put it on top of the stack. We then take another plate, write a two on it, and put it on the pile. We add a plate with a one on it to the growing heap, and then finish off with a plate labeled zero. Now, when we remove the plates, the zero plate comes off first, followed by the one, then the two, and finally the number three plate. Note that the order in which we remove them is exactly reversed from that in which we placed them. Note also that there is only one number visible at any time, even though all of the numbers that are beneath it (placed on earlier) are "remembered." In our factorial function, we want N to take on successively the values 3, 2, 1, and 0 and then take on the same values in reverse order (1, 2, 3) as it returns from successive function calls. Going back to the plates, note that we placed plates on the stack, then removed only what we put on. There could have been plates underneath; if so, then they would not have been disturbed. This process is useful because our factorial function may not have been the only one called. For example, a function to compute sines may have been called, which then proceeded to call this one. This "stack of plates" concept, while it might not work, seems to give hope for finding a way to do recursion.

How do we write something in BASIC that can use this? There are two problems: interfacing to a stack of plates is not easy and it would be a very slow peripheral since we have to wait until someone comes to eat in order to see what is under the top plate. Besides, people might complain about numbers being written on their plates. Thus it seems reasonable to write a simulation of what is going on that will run using only BASIC and needing no fancy peripherals. What better object to use than an array, since we probably will wind up doing the same task over and over on many different values. We will start off by being completely arbitrary and do the following: DIM N(100), R(100). What, you ask, are we going to put in array N? N will contain the numbers that would otherwise have been written on the plates (3, 2, 1, etc.). Its use is to keep track both of what is in current use and the values to be remembered. R is the array that notes which line we were called from. It is being kept track of internally by BASIC itself and is being shown here to help you follow what is going on. (Remember that this BASIC can handle recursion.) Let's see what might be in N and R immediately after we call factorial for the first time:

N(1)=3 R(1)=666 (The function was called from line 666)

(Assume that any elements of the arrays N and R that are not specifically mentioned contain garbage.) We will now continue until FNF calls itself for the first time (the second call). Now, N and R look like this:

N(1)=2 R(1)=50
N(2)=3 R(2)=666

Until now, there has been no question about what the value of N is. Now, there are two entries in the array and there must be an agreed-upon method for figuring out which is the most current one. (Such an agreement is called a convention, and few, if any, conventions are industry-wide.) Judging by what we did to get into this situation, we shall adopt the convention, temporarily, that the currently active element is N(1). Thus, whenever we encounter an N in the function, we use the value in N(1) to tell us the current value for N. When each number is placed on top of the stack (this corresponds to making another copy of the function; this is called *pushing down* the stack), all of the other values can be moved one slot further down to "make room." As we shall see later, removing the top value (returning from a function, called *popping* the stack) allows all of the buried values to be moved one step towards the top.

We will continue executing the function and stop when N is zero and FNF has just been set to one (for the first time).

N(1)=0 R(1)=50
N(2)=1 R(2)=50
N(3)=2 R(3)=50
N(4)=3 R(4)=666

If you feel a bit lost now, go back and look at the expanded diagram (Fig. 1). We are currently on line 30 of the most deeply nested copy of the function. Let us continue on until we are just about to return from the copy of FNF where N=1 (line 50 of the next-most-deeply nested copy).

N(1)=1	R(1)=50
N(2)=2	R(2)=50
N(3)=3	R(3)=666

Note how the stack has popped. It should be apparent by now what will occur when we let execution continue until we return to line 666. Both of our problems have now been solved and the "stack of plates" works. If this array structure is incorporated within the BASIC system, then there no longer need be more than one copy of the function. We also can handle the statement LET N=5* because we can modify N(1) all we want, since it will be thrown away upon returning.

There still is a very inefficient process going on here. So far, each time we push something onto the stack, we laboriously move each entry down one place. This is very wasteful of time. This copying is done because we want the current-use position to be constant (the first element) but we can let the remembered values move about. The trouble is that there are usually more remembered values than current ones. So let's flip the stack end-over-end and fix the position of the remembered values, while allowing the position of the current one to change. We have to keep track of this position, so we shall introduce the variable P (for pointer, if you really must know) to do this.

The previous four snapshots will be reprinted in the new format:

N(1)=3	R(1)=666	becomes	N(1)=3	R(1)=666	P=1
N(1)=2	R(1)=50	becomes	N(1)=3	R(1)=666	P=2
N(2)=3	R(2)=666		N(2)=2	R(2)=50	
N(1)=0	R(1)=50	becomes	N(1)=3	R(1)=666	P=4
N(2)=1	R(2)=50		N(2)=2	R(2)=50	
N(3)=2	R(3)=50		N(3)=1	R(3)=50	
N(4)=3	R(4)=666		N(4)=0	R(4)=50	
N(1)=1	R(1)=50	becomes	N(1)=3	R(1)=666	P=3
N(2)=2	R(2)=50		N(2)=2	R(2)=50	
N(3)=3	R(3)=666		N(3)=1	R(3)=50	

In the first example, nothing much is accomplished. However, in going from the first to the second, no moving around was needed but only the incrementing of P and the placement of new values. In going from the third to the fourth example, instead of having to move up three entries in each array, all that happened was that P was set to P-1 (decremented). (Elements N(4) and R(4) don't even have to be erased. They just become part of that garbage that we mentioned earlier.) The only price for this increase in efficiency is that you must use N(P) instead of N(1) whenever you want to refer to the current value. Note that in neither case does it make a difference whether or not there was already something on the stack (P might just as well have been, say, ten to start).

The size, 100, which was picked arbitrarily, actually does have some significance. When you select the stack size, you must be sure to allocate enough room for the largest number of items to be remembered that you will ever come across. This can lead to having large amounts of empty space tied up in stacks that is not likely to ever be used. The common solution to this is to combine the various stacks into one large stack. For our example, we could do something like the following:

P=2	S(1)=666	the entry from R
	S(2)=3	the entry from N

To put the next function call on the stack, the 50 (from R) would go in S(3) (the odd element) and the 2 (from N) would go in S(4) (the even element). The main difference is that P would now go up by 2. The effect of all this is to have one array of size 200 instead of two arrays of size 100. Now, 2x100=200 and you might ask, where is the savings coming from? The space savings doesn't start until the things to be stored get more complicated. This savings comes when two or more stacks being combined do not always grow at the same rate. An example might serve to illustrate.

Function calls are the only things that we have been putting on the stack so far. There is another class of related things that also is convenient to put on the stack when writing an interpreter. This class is mainly concerned with evaluating expressions, and consists of open parentheses, subscript calculations, and operands for operators that have been delayed in execution (that is, in 4+5*6, the 4 must be held somewhere while the 5*6 is evaluated). Function calls can occur anytime and so can open parentheses, etc., but they occur independently and the occurrence of one does not imply the occurrence of the other (with the exception of the parentheses around the function's arguments). Usually, there are many more open parentheses, etc., than function calls.

Since we have two things which occur independently, we can make good use of the "common extra space." If there is a sudden rash of function calls, the common space can be used to handle it. The same storage, at a different time, can also be used for open parentheses, etc. Thus, we don't have to dedicate it to any one particular use. (The situation can arise where we want to call a lot of functions *and* evaluate a complicated statement. Here, the only solution is more storage.)

The entries on the stack containing the function calls or open parentheses will probably be of different, if not varying, lengths. An example of a varying-length entry is a function call, since it can have anywhere from zero arguments on up. Thus, it might prove useful to keep track of the lengths of the entries. In general, it somehow makes sense to place this length at the start of the entry (start being defined as the first part of the entry that you see). Later on, there will be a reason given for keeping those lengths around.

Believe it or not, you now know enough to implement recursive functions in your BASIC system, either by modifying the system itself or more simply by simulating. The main difficulty—in either case—is figuring out how to handle R (the return addresses), since it is that which most systems lack in a readily usable form. This is part of what APL can do that most BASICs do not.

Variable Creation/Deletion

Already we have seen how APL could do recursion, and that is the first of the differences mentioned. Variable creation/deletion "on the fly" is the other difference mentioned. We will start by examining what exactly it is that we are creating and deleting. Since we are talking about variables, it seems logical to start by keeping track of their names. APL, unlike BASIC, offers multi-character variable names, so it seems time to apply the length concept and store the number of characters in the name followed by the name itself. Another logical thing to do is to note down the value of the variable. Since APL allows multi-dimensional arrays, the number of dimensions (whether the name represents a simple variable, an array, a matrix, etc.) should be stored, followed by the length of each dimension (another application of the length concept) followed by the value(s) itself (themselves). Alternatively, there could be "flag values" telling when, for example, the dimension list ends, eliminating the need for a number of dimensions entry. There are additional ways of doing this. Note that functions can be stored in a similar way, so the later discussions can be applied to either. Since what we have here amounts to a variable definition (as in definition of a variable), that makes as good a term as any to use when referring to the whole thing. The variable definition itself, of course, can have a length associated with it.

---

*This method allows us to do what we mean in a recursive program. When we have a function call itself (that is, FNF(N-1)) what we mean is that N in the new copy should have a value one less than that in the calling copy. This is accomplished by N(1)=N(2)-1 in the calling sequence. Now, modifying the current N will not affect the old values. A variable that has this property is said to be local to the function.

To keep track of all the variable definitions, we will store them in a table, simply listing them one after another. (Needless to say, this method will be replaced with a more efficient one!) Now, the table looks like this:

variable definition #1
variable definition #2
variable definition #3
•
•
•

To add a new variable definition to the list, you simply tack it on the end. To delete one, you could just replace it with zeros or some other marker. (This same marker could fill in the rest of the "scratch area" that this table is stored in.) There are three ways of modifying the definition that are of import here: making the new definition (1) shorter, (2) the same length, or (3) longer than the old one. If you make the new definition the same length, there is no problem. If you make it shorter, you can fill in the extra space with the marker value. If it is now longer, you can fill in the entire old definition with the marker and tack a new definition on the end.

This method is undesirable for three reasons. First, by always moving large chunks of data around, gaps tend to develop very quickly. Second, while this method allows recursion, it requires much too much shuffling of data to be efficient. One major inefficiency is that when figuring out which N to use, you might use the first one that you come across. This means that, when calling a function, if there is already an N in the table, you must insert a *new* definition such that the new definition is found first when searching the table. This entails pushing the old definition to the end of the table and replacing it with the new one—perhaps only temporarily. And then there is always modifying the new definition .... Last, this makes it nearly impossible to enforce order (alphabetic, for instance,) on the table.

The first problem can be solved by breaking definitions into smaller chuncks. Whenever a chunk has to be modified, only that part need generate a gap. This tends to keep the gap-size small, and, by not duplicating data unnecessarily, uses up storage at a slower rate.

The second problem is somewhat more complicated. It really gets emphasis when you are searching the table to find a name. To do this, you must find a name, then go element by element until you find a marker, then element by element until you find the next name. Needless to say, this is very slow. For an improvement, remember those lengths that we insist upon hanging on to without giving a reason? Instead of having to search through the whole table item-by-item, you can use these lengths to skip over the variable definitions entirely. Now, all that you have to do is search through marker values to find the start of the next name entry. And, if you were to hang on not only to the current lengths but also the original lengths (before shrinking), you could skip over the whole area and go directly from name to name. You now have a linked list on your hands (don't worry—it isn't contagious). Simply stated, a linked list is one in which you have a data block, with a pointer to the next data block, etc. (See Figure 2.)

This, however, is still not the best of all possible worlds. When definitions get deleted entirely, it can become very difficult to keep the table correct. This can be taken care of, in effect, by forgetting that the deleted variable was there at all. This can be done by causing the pointer which pointed to the variable-to-be-deleted-now to point to the next entry. Inserting an entry is just the opposite case. (See Figure 3.) Moving an entry is easy; you just cause the pointer to the entry to be moved now to point to the new location. (See Figure 4.)

This leads to the interesting prospect of the next name not having to be anywhere in particular. In other words, when you have to move a definition because it no longer fits in its present slot, all that has to change is the pointer in the preceding entry and, as far as the list is concerned, nothing has happened. This nicely solves the third problem in that it is now trivial to keep the list in alphabetical order (or any other order, for that matter). Also, recursion can be supported much more easily because the new definition can be added to the front of the

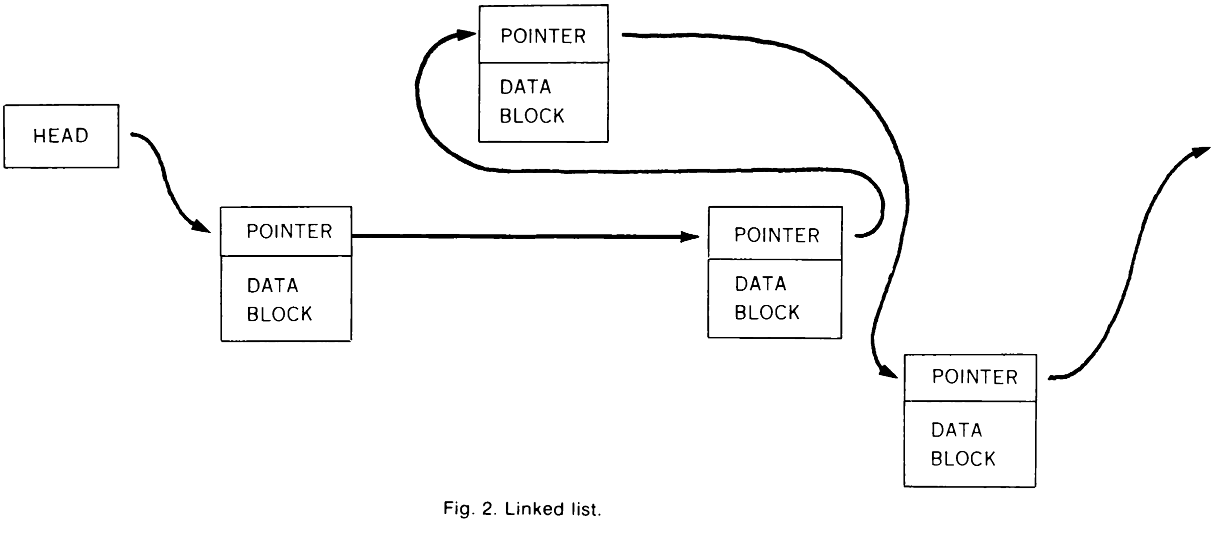


Fig. 2. Linked list.

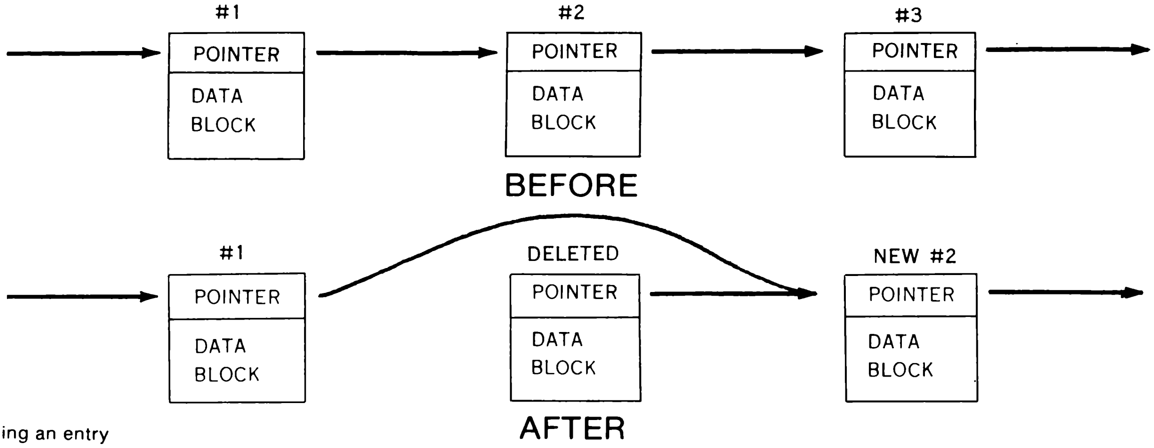


Fig. 3. Deleting an entry

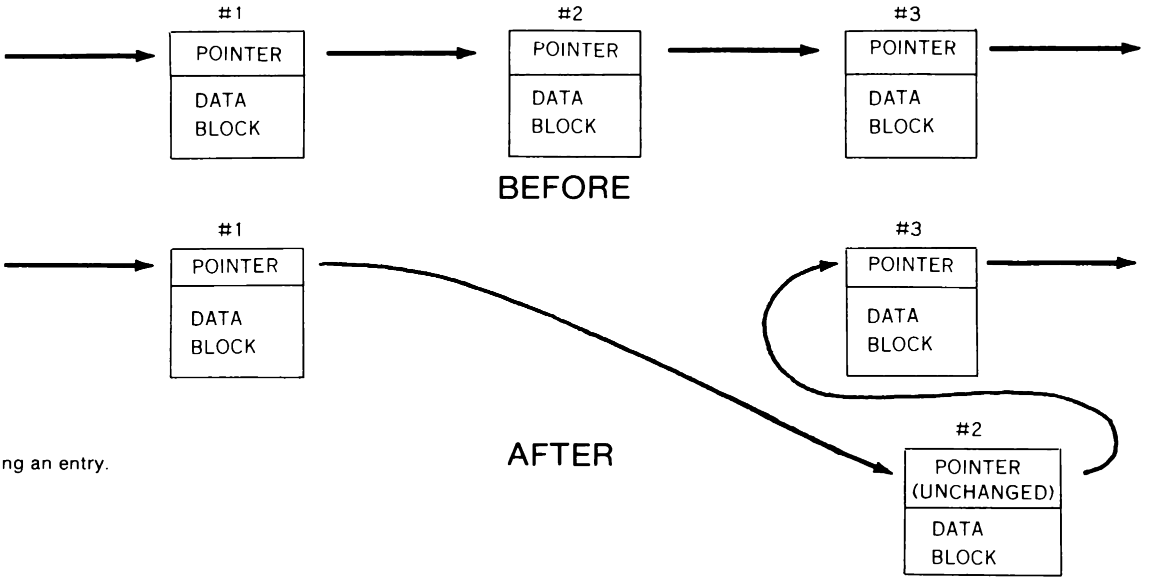


Fig 4 Moving an entry.

list, even though it might be located at the physical end. There is still one minor flaw in this idea, and it is tied up in what the garbage-collect (described later) does. The salient point is that the garbage-collect moves stuff around, and the next entry is relative to where you are now. Thus, there are two ways for the pointers to get messed up. The most obvious change is to use absolute (relative to some fixed point) locations instead of relative (to where you are now) locations. Now, the pointers only have one way to get messed up. (Besides, by using absolute locations, each entry becomes independent of where it is located and can be moved around more easily.)

Although at first glance there doesn't seem to be any way of stopping the lists from going on forever, there's a convention regarding this which is to use an illegal or unused address as the final pointer (often zero). By checking for this address, you can find out when to stop.

It should begin to be apparent that when storing information like pointers, lengths, and other such things, we are using storage for non-data items. This is part of the trade-off between low overhead (amount of space used for "housekeeping" purposes) and flexibility in the system. This means that if you only want the data, then you are very limited in what your options are, while if you want to keep a "description" of the data, you have more options and even—to some extent—the ability to ask questions. In a simple example. you can write a program to average ten numbers, or you can write one to average N numbers. In the latter case, you have overhead in that you have to store N. You also have an option in what you can do and you can even ask what N is. The same concept applies on a much larger scale when you start keeping as many extras as we are. There is a different trade-off: that between low overhead and low execution time. The extra information requires space, but it significantly cuts down on the time needed to execute a program.

Garbage-Collect

Finally, there is yet another toll exacted when programs become dynamic (constantly changing). Remember all those gaps that are created whenever you update a definition (or almost anything, for that matter)? These accumulate until a sizeable fraction of storage can be eaten up by them. They must then be packed down from time to time to keep the program from running out of space while there is still unused storage sitting around. It is this packing down that causes all of the lists to be moved around. The program that does this is called a garbage-collect and must—unfortunately—be individually written for each application. Garbage-collects are another use for all of those lengths, since it is convenient for the garbage-collect to know how long a chunk of data is in order to move it. Although this requires extra space to store, it saves large amounts of time and even cuts down on the size of the garbage-collect routine. It should be noted that there are ways of avoiding a garbage-collect. These involve having fixed-length chunks of data (to eliminate small gaps) and keeping a linked list of unused space. This method is often used in managing disk space, since garbage-collects on disc are very time-consuming.

Name Table

Turning back to our table of variable definitions, we find that it now could contain only the names of the variables. In the entry for each name might be a pointer telling us where the rest of the definition is. The rest of the definition may even be sitting right next to the name, but that doesn't matter. The reason for this is that when we are searching the table, we are only interested in the names until we find the correct one. Only then do we look at the rest of the definition. Hence, from now on, we will call it the name table.

The name table, along with the stack containing information on function calls, etc., can tell us everything that is going on in and around the program. This, then, is called the environment and, even though the details or even the major parts can change from program to program, the term environment still refers to the "general picture" of what is going on.

Our name table is a linked list which has the names in some order. The only ordering that we will assume is that when a function is called, its parameters' (arguments') definitions will be placed in the front of the list so that we will find them before their earlier definitions (either in the main program, another function, or another recursion level), if there were any. When a function finishes (returns), the most recently added definitions (that is, those that it added— isn't it nice how stacks work?) are removed. Thus, parameter definitions only exist after a function has been called and before it finishes. Thus they are local to the function and are called local variables. (In almost all languages that support recursion, there is a way to define a variable as local without its having been passed as a parameter.) The way then to determine which definition of a variable to use (assuming that there is more than one definition) is to start at the beginning and use the first occurrence of the name.

This method is reasonably straightforward, but it does have one drawback. If you are calling a function that has five local variables and this function calls itself ten times, then there are fifty entries in the table followed by whatever was there before. There will probably be two types of variables that you are looking for: local variables (the most recent five of the fifty) or anything else (what is after the fifty). It seems rather inefficient to have to look at forty-five names without a hope that what you want will be there. Now, the situation on the stack will change whenever we call another function or return from one. This happens a lot less often, on the average, than trying to find the definition of a variable. We can, therefore, afford to spend a little extra time doing some polishing up of the name table at that time. The polishing up, of course, consists of effectively removing the entries of those forty-five "extra" definitions.

When we put a function call on the stack, we also could put a listing of the local variables for that function with the call. This is useful in telling us which variables to delete from the name table when we return. Suppose, for the moment, that we would also store a pointer to the definition for the variable that it replaces (or, say, a zero if there wasn't one). This eliminates the need for the old entry in the table and it can be deleted. We now have only the new entry in the table and, in the function call, a pointer to the old entry. (See Figure 5. The function F has local variables A and C. There is only one entry for A in the name table, yet all the information can be recovered.) Thus, there is no information lost and upon returning, it is a simple task to restore the table by replacing the pointers to the new entries with those of the old and deleting all names that have our marker (zero) in their function call entry.

This concludes the introduction to some ideas, concepts, and "tools of the trade." Of course, many details were omitted, but then these vary greatly depending upon the exact nature of what your program is to do. You should now be able to see how APL—or any languages that are dynamic and/or allow recursion—put these techniques together to come up with a working system. ■

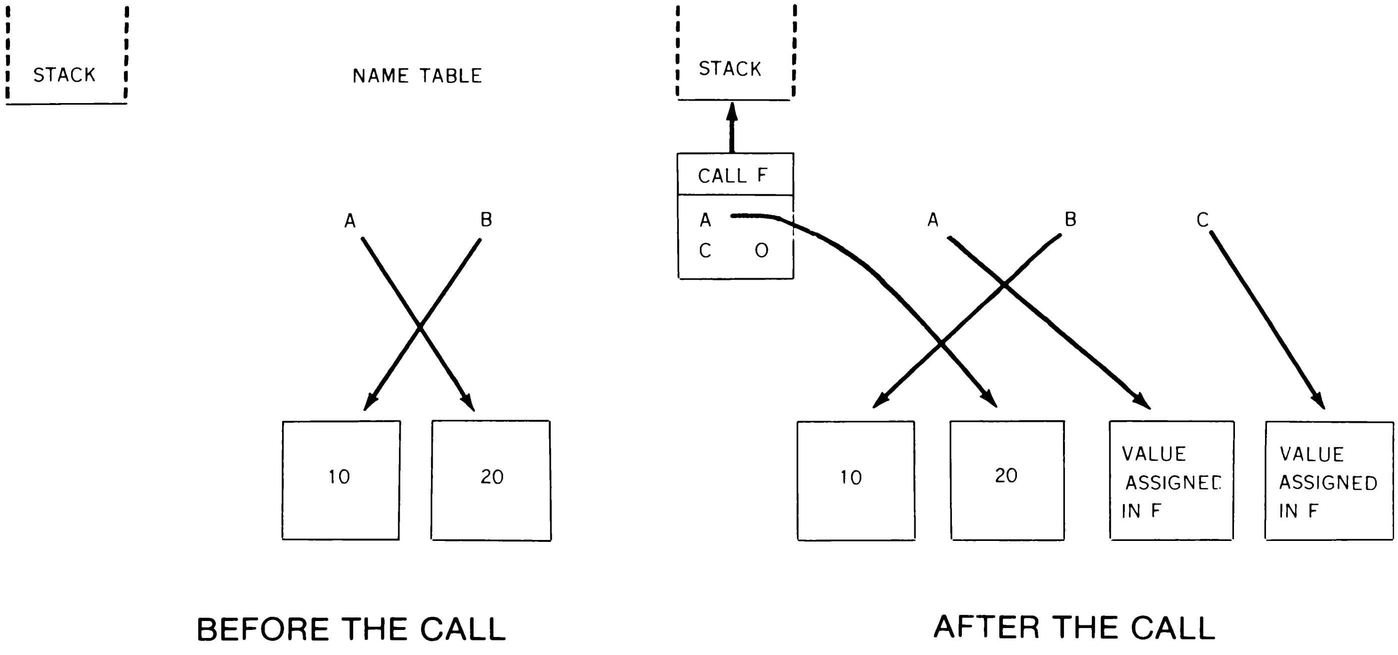


Fig. 5. Stack and the name table.

Programming Techniques: File Structures

John Lees

The primary use of general-purpose computers is data processing, and most of data processing is file handling. To make effective use of your computer, sooner or later you'll have to deal with files. So here's an introduction to the wide world of file structures.

To begin with, a few definitions: A *file* is an organized collection of related information. (A collection of files can be called a data base.) A file consists of *records,* all of which usually have the same basic structure. Each record can, in turn, be subdivided into *fields,* or *elements.*

A file can exist in memory, on paper tape, cassette, magnetic tape, disk, or any other type of storage. Storage can be considered in two categories, that allowing only sequential access (all kinds of tape and some primitive disk systems); and that allowing random access (semiconductor or core memory and most disk systems). Some tape systems purport to have random access, but random access on tape can only be achieved with considerable access time and space overhead.

Sequential File

The simplest and most common file structure is *physical sequential.* Records are arranged in some order, one after another in the file, and are physically accessed in that order. Tape files are by their very nature physically sequential.

Let's consider an example of a sequential file. Say you want to keep a catalog of all the books in your library, or all your record albums, musical scores, paintings, or any similar item. A record in such a file might look like this:

AUTHOR	TITLE	PUBLISHER	ADDRESS	BINDING	PAGES	PRICE	DATE

The file would consist of one such record for each book in your library. Probably the order in which you would choose to keep such a file stored would be alphabetical by the author field (which would be the *key field* for the file), but of course you could store it in any order you wish. The order you decide to use is important though, since you don't want to have to sort the file each time you use it.

Once you set up your library file, you'll want programs to add books or delete books and possibly to allow you to modify a record, although you could get by with deleting, then adding, to modify. You'll want the capability to print, say, all titles by one author, or by one publisher, or all hardbound books, or all books published in 1974.

Two Transports

If you're using tape, you'll need two transports to be able to keep the file in order, when you add or delete records. To do that, you'd write a program to take all the additions or deletions, sort them and keep them in memory. Then your program would read from one tape, writing records out to the other tape until it reaches the point for an add or delete. It would do the add or delete and then keep on reading/writing until the entire file had been processed.

You may have noticed that a lot of space is being wasted in our sample file. There is only a small number of different publishers, yet that information is repeated in every record. Very wasteful! To save space, we could create two files, one file consisting of a modification of the records we already have, with the publisher information replaced with a code:

AUTHOR	TITLE	P-CODE	BINDING	PAGES	PRICE	DATE

and another file containing the publisher information:

P-CODE	PUBLISHER	ADDRESS

These two files could be used in this way. Put the Publisher file first on the tape, followed by the Library file. As the first step in using the Library file, read the Publisher file into the memory since it is relatively small. Now, as records are read from the Library file, the publisher code can be matched up with one in the Publisher file and the information in the record in the Publisher file used to print the book listing. This look-up will be fast since it is done in memory. The small amount of extra processing time used is well justified by the savings in file space. This same principle can be applied whenever a field contains often-repeated information. Of course the information must be longer than the code used to replace it. It wouldn't pay to do this with the date field, for instance.

Tight Space

If you're real hard-up for space, the leading "1" need not actually be stored in the date field. Similarly, don't store a "$" or even the decimal point in the price field. Have the program add them when it adds the publisher information to the book listing being printed. You can save even more bytes by storing the price and date in binary and converting. But don't get carried away if you don't need to save the space.

All well and good, but what if you have collections, such as science-fiction anthologies, and want to be able to find authors and titles of stories in the collections? You could do about the same thing we did with publisher information. Add a code to the author field:

AUTHOR	B-CODE	TITLE	P-CODE	BINDING	PAGES	PRICE	DATE

This new code could mean if 0 then the book is not a collection, else the code would match up with a set of records in a short-story file which would give the contents of that collection.

But that isn't really what you want. That scheme will let you list the contents of a collection, but is of no help in finding out if you have in your library a short story that only appears in a collection. What to do? You could, instead of having a separate short-story file, include these records in the main file. Avoiding redundant information, you would now have a file consisting of three different types of records:

TYPE 0	AUTHOR	TITLE	P-CODE	BINDING	PAGES	PRICE	DATE

TYPE 1	B-CODE	AUTHOR	TITLE	P-CODE	BINDING

TYPE 2	B-CODE	AUTHOR	TITLE

A type 0 record would be a normal book. A type 1 record would be a collection and such a record would contain an additional field with a book code, which would match a book code in type 2 records containing the authors and titles of the short stories in the collection. To make this file easily usable with a sequential storage medium, you'd probably want to group all the records together by types (almost, in effect, giving three files), in alphabetical order by author within type.

Now if you want to see if K is in your library, the program would look for a record of any type with K in the author field. If the record(s) found with K are of type 2, then the program would also look for a type 1 record with matching b-code and tell you what collection the story by K appears in. If you think that this kind of thing could take a very long time on a cassette, you're perfectly correct. But what's your hurry?

Faster, Faster

Well, maybe you're writing this system for the school library and you have a legitimate reason for wanting the search for a book to take less than half an hour. Hopefully you can get a disk or two, because you've exceeded the capabilities of a sequential-file structure. The rest of the structures we're going to discuss require random-access devices.

The drawbacks to the plain old sequential file are obvious. If you're on Heinlein, you know that Vonnegut is somewhere further on and that you've passed Ellison. Vonnegut you'll eventually come to, but Ellison can only be reached by going back to the beginning and starting over again. An unhandy state of affairs if speed is of any importance at all.

Index

So, enter the next bright idea in file structures, the index. Although the combination of indexes and sequential files will not, as IBM once tried to convince the world, solve all problems, it does help a little. Imagine a dictionary with no way of telling where each letter begins and you'll quickly appreciate the utility of an index. The idea is the same with a file. You have the master file and an *index file* which contains the information on where certain categories begin in the master file. This could be in terms of record number, memory address, disk sector, or (shudder) tape block. Now if you want to find Heinlein, the program looks in the index file and goes right to the beginning of the H's. This is of limited use on tape, since you still have to move all that tape past the read head slowly enough to count blocks.

With our example, you could also have an index to help find the groups of short-story records and even the records for the collections themselves. So your program could go right to the record or group of records. A diagram for such a file system might look like this:

Alphabetical Index

A	LOCATION

•
•
•

Z	LOCATION

Short Story / Collections Index

B-CODE	LOCATIONS OF SHORT STORY RECORDS	LOCATION OF COLLECTION RECORD

•
•
•

B-CODE	LOCATIONS OF SHORT STORY RECORDS	LOCATION OF COLLECTION RECORD

Library Master File

TYPE 0	AUTHOR	TITLE	P-CODE	BINDING	PAGES	PRICE	DATE

TYPE 1	B-CODE	AUTHOR	TITLE	P-CODE	BINDING	PAGES	PRICE	DATE

TYPE 2	B-CODE	AUTHOR	TITLE

Publisher File

P-CODE	PUBLISHER	ADDRESS

Once you've figured out that conglomeration, you'll see that it saves a lot of work, at the expense of a little storage space. Using the indexes may speed things up, but a lot of sequential processing is still required, and the records within groups must still be kept in alphabetical order, thus requiring a lot of insert overhead. (Deletes are simple. Just adopt the convention that a type 3 record isn't there and so mark "deleted" records, every once in a while collecting the garbage and squishing things together.) Also, there are a couple of little bugs in that file system and a very high maintenance cost associated with updating all those index records if any of the Master file records are moved, as they will be each time an insert is performed.

It is possible to get away entirely from any reliance on sequential ordering, at the expense of a little more storage and a little more processing time. But processing time is cheap and maintaining a sequential file is a nightmare when you don't have much memory. So let's move on into the realm of list structures, linked lists, rings, trees, hierarchical files and such things. You ain't seen nothing yet! ■

PILOT

Gregory Yob*

PILOT is a dialog-oriented interactive language for use by teachers and students on small systems. Its simple syntax and free format encourage innovation and use by those frightened by computers or who lack time to learn a more complex language.

PILOT—A TOOL TEACHERS CAN USE

One of the hidden factors in introducing new technology to the classroom is its demands upon the teachers. A teacher's time is quite limited as it is (with state requirements, meetings, etc.) and any new technology or methods should increase the net time for teaching.

Bringing the computer to the classroom usually complicates matters. A typical CAI system forces the teacher either to learn a complex language or a fixed and complex curriculum package. In both cases, the time required for skillful use of the computer is usually too great for effective utilization.

The usual result is that skilled programmers prepare vast and inflexible curricula which are then given to the teacher. This is horribly expensive and inefficient. What is needed is a means of generating materials immediately, quickly and simply for the day-to-day requirements of the teacher. The tool must appear "natural"; that is, it must look like natural-language dialogues; it must allow for variations of style, nuance and tempo; it must appear riduculously simple; that is, be "learnable" in less than an hour. It must avoid the computeristic scientific bias which separates the math/sciences from the English/humanities areas of teaching.

The PILOT language is the beginning of such a tool. Its basic four functions (T: to type text; A: to receive an answer; M: match keyword; J: jump) can be taught to nearly anybody in ½ hour. this includes persons who just won't (can't) understand math and stuff like that.

PILOT programs are written as simulated dialogues in English (or Spanish or... .) and can be entered and executed quickly. Brief programs for special purposes are easy to do and since the teacher is doing it, not the curriculum designer, the program is just as easily changed or discarded.

The dialog format of PILOT also allows immediate understanding of PILOT programs, making them highly exchangeable with other teachers, and encouraging the dissemination of good ideas. This is in contrast to most other computer languages, which require a detailed description of the program as well as the code.

These features make PILOT a viable and non-time-consuming tool for teachers using computers in education.

HISTORY OF PILOT

PILOT was developed in 1969 at the University of California Medical Center by John Starkweather to meet some instructional needs. It was used to train students in pharmacology and later in an elementary school in Marin county. Stanford Research Institute used PILOT in an experimental educational research project (with very good results) and later developed a dialect, Called PYLON, which was a very simplified version. In 1971 and 1972, other variants of PYLON were developed by Stanford University, The California State College computer network and Lawrence Hall of Science at the University of California, Berkeley. A small version of PILOT was made by John Starkweather for stand-alone operation on the Datapoint 2200.

In January, 1973, the varied users of PILOT and PYLON met to standardize the language. A standard "core" version was agreed upon, called PILOT 73. The "core" version includes standards for extension as each user is free to make his PILOT more powerful for his system.

Currently, PILOT is implemented in about a dozen languages on 20 or so systems. There are 25-30 sites actively using PILOT nationally at present. A user's group, called "The PILOT Information Exchange," disseminates information and initiates contact among those interested in this language.

---

*Gregory Yob is an author of several versions of PILOT and has participated in the definition of the language. He currently coordinates the PILOT Information Exchange, a national user's group for PILOT. Other areas of interest include computer games and working with neighborhood computer centers.

THE RELATIONS OF PILOT TO OTHER LANGUAGES

The thrust of computers in education seems to be mostly in these areas: First, courses designed to teach about computers, how they function, and how to program them. Second, using the computer to pass curricula or other study materials to the student. Third, actively involving the students in using the computer to solve problems in their course of study. fourth, allowing students to use the computer for their own expression, self-integration and growth.

Each of these areas have languages associated with them. FORTRAN, assembly language, COBOL and BASIC are taught with courses concerned with how to use and understand the computer. COURSEWRITER, LYDIA, TUTOR and PILOT are used to pass courses of study. Most problem-solving is done in BASIC with small efforts in FORTRAN, LOGO, and SMALLTALK. PILOT and LOGO are often used for self-expression and growth.

That's how PILOT fits into the general picture. A closer look at four languages will give a sharper focus:

COURSEWRITER is designed for presenting course material to a student. The teacher, or more often, the curriculum developer, is expected to write series of lessons, tests of the student's progress, etc. The student is exposed to the material being taught, and NOT the language. Unfortunately, (if you ignore the salesmen), COURSEWRITER is too complex and inflexible for most teachers to use effectively, which leaves the development of materials in the hands of specialists. As teachers are unable to provide feedback, the proffered courseware packages lack vitality and are often little more than mechanized textbooks.

BASIC is in essence FORTRAN with a lot of garbage removed. Its ready availability on small systems (especially timesharing systems) makes it quite popular for math and science teachers. The student learns BASIC as a tool for solving numerical problems posed in his courses. However, BASIC is very weak with strings and words. The humanities staff has never heard of BASIC and even BASIC's relatively simple syntax is too difficult for the word- and English-oriented person. The result is that BASIC users tend to be in the scientific and technical disciplines.

LOGO (like BASIC vs FORTRAN) is a simplified variation of LISP with control of devices other than the Teletype—such as the "Turtle," a plotter-robot or "Music box," a tone generator. LOGO is self-extensive and capable of handling lists and recursive function calls. LOGO is excellent for problems concerning the order and arrangements of things—procedures rather than calculations. At present there are few users of LOGO and they are mainly math- and computer-oriented.

PILOT is a dialog-oriented language which deals mainly with words and text, The syntax is extremely simple, allowing teachers and students to learn it readily. Because much educational material is essentially in English, non-mathematical users find PILOT a flexible tool for presenting materials via the computer. PILOT lacks arbitrary conventions such as "Frames" (viz IDF)* and counters on every answer by the student which are often unnecessary and hamper the style of the program author.

It is clear that PILOT will not and is not intended to replace the other languages used in educational applications. It fills a complimentary place among dialog, numerical, procedural, and curricular languages.

THE PILOT LANGUAGE

Now for the part you have all been waiting for—what does PILOT look like? Here is a very brief introduction to PILOT. Contact the Exchange for a more detailed description. This program almost teaches you the language—see if you can figure it out first.

```
      T:Hi there. Is this your first time on a computer?
      A:
      M:yes,sure,ok,yeah
      TY:I hope you will enjoy your experience with me.
      T:In the area of education, what are your main in-
        :terests?
*MORE   A:
      M:teach,instr,leam,material
      JY:*TEACHING
      M:admin,program,test,grad,analys,course,curr
      JY:*OTHER
      T:Please tell me more about this.
      JN:*MORE
*TEACHING T:An excellent way of using computers for
teaching and learning
      :is to give children an opportunity to write their
       :own programs.
       :How does this strike you?
      A:
      M:good,excel,fine,yes,important
      TN:I see that you disagree. Will you explain fur-
       :ther?
      JY:*YES
      A:
*YES T: Of course the teacher should write programs
too. However,
      :it isn't always necessary to use "packaged cur-
      :ricula" for effective use of
      :the computer in learning situations.
      E:
*OTHER T: Are you interested in the computer's ap-
plication to teaching?
      A:
      M:no,never
      JN:*TEACHING
      T:Then perhaps PILOT is not for you. PILOT's in-
      :tention is for its use by teachers
      :and children for interactive dialogues. Thank you
      for your time and interest.
      E:
```

A look at this program shows four basic kinds of statements:

T: This means to type out the text following the colon (:).

A: Here the computer stops and awaits a reply by the user. Answers may be saved for later use in T: statements by following with a $-variable. Here is an example:

```
T:Who are you?
```

---

*Hewlett-Packard's CAI "language."

```
A:$NAME
    T:OK, $NAME, I have a puzzle for you.
```

M: The last reply is examined for the keywords in the list following the colon. If there is a "Match," any statements with the Y (like TY:) will be executed. In the case of no match, the N-suffixed statements execute.

J. Jumps to any part of a program are possible. Branching is essential for differing presentations according to answers given. The *label is a tag indicating where to go. JY and JN are often used to vary branching on replies.

Successive applications of the M: can perform precise analysis of an answer. M: also allows the search of words, suffixes, prefixes and text fragments by allowing the blank as a legal match character.

Once a M: is executed, the yes or no (Y or N) is effective for any statements with the respective suffixes. Statements lacking Y or N will always execute.

The ability to save answers allows the simulation of intimacy and personality by careful use of echoing. Amusing stories, poetry, etc. are possible in this manner. The personal-seeming responses are very important for the captivation of interest.

Some more advanced statement types are:

R: Remarks for documentation
C: Perform computation (usually in host language, like BASIC)
U. Call a subroutine tagged by a *label
E: Return from a subroutine or program end.
A:#letter Allows numerical variables which can be modified in C: and presented in T: in a manner similar to $variables.

PILOT has protocols for extensions peculiar to the system it is running on. For example, a useful extension may be:

```
SCREEN:UP 5
```

This would move the cursor up five lines on an alphanumeric CRT with cursor control characters.

When PILOT is implemented in a higher level language, such extensions are easily made.

This extremely simple syntax and consistent form lets the program writer concentrate on the quality of the dialogue and important branches rather than the picky details of syntactic form. PILOT is very rugged and can tolerate sloppy code which is encouraging to the beginner. Very elegant and complex programs may be written if needed.

GETTING PILOT ON YOUR SYSTEM EFFECTIVELY

Getting PILOT really running on your system requires more than the language processor.

This section describes an ideal which is seldom met in practice. The author notes that the successful users of PILOT usually do most of the following, sooner or later.

SOFTWARE

It is an illusion that having PILOT will do it all for you. There are systems in which loading a program is harder than mastering the language it is written in!!

PILOT. Obviously you must have some kind of PILOT. If you have a choice of versions (as many do), points to look at include: speed and response time, use of mass storage—efficiency and number of accesses, completeness of PILOT 73 (the standard version), the ability to tolerate badly written programs or at least to try hard at running them, idiot-proofing (no way to crash the program or system, even intentionally), clear concise error messages, availability of documentation and technical help.

EDITOR. Somehow the PILOT programs must get into the machine, saved and changed. At text editor is often used for this, though some versions of PILOT include an editor/syntax checker. Beware!! Recently at San Francisco State University, the system's text editor was so very complex that the PILOT was never used. The problem was cured by writing an editor especially for the PILOT. There is no point to a language simpler than its editor!!

The trend in PILOT editors is to emulate the BASIC line-by-line editor. This is usually wise, as many PILOT users will go on to BASIC or vice versa. The editors which use pointers have been hard to teach to children using PILOT. The eventual cure is to have a full graphics system with joystick or mouse for editing. (May the day come soon.)

TUTORIAL OF PILOT. As PILOT is an instructional language, apply recursion and use PILOT to teach PILOT to new users. Many new users will prefer to learn at the terminal, so writing this package is a good checkout of the system and teaches PILOT to that first group of users.

A good tutorial is a system in itself, and these features may be included. (A) Help and summary information. Who to call in distress, A list of editor commands, a list of PILOT statements, a list of options within the tutorial system. (B) The PILOT tutorial set of programs with a reference page or two for the experienced user (C) The editor tutorial—don't forget this one! (D) How to use the %¢&&% terminal.

The seed library can be very helpful in setting the style and mood of PILOT's use. Many teachers will refer to the seed programs as models for their teaching work. Without this stimulus, the teachers will fall back on their preconceptions of computer-based teaching, resulting in the dull boring drill or curricular approach to PILOT. A PILOT user submitted to the Exchange some very long programs which were entirely questions and multiple-choice answers—having never seen a light, fun PILOT program. This point cannot be overstressed. Without new models, old ones prevail, despite their inappropriateness for current needs.

To summarize: the PILOT language, the Editor, the PILOT tutorial system, the Seed programs, are all important software.

REQUIRED DOCUMENTATION

Many of these follow closely the items mentioned in the software section. The only effective way to make documentation is to insist on it BEFORE anything else. Documentation is almost always done after the fact, with parts missing and a generally reluctant attitude. Good luck!!

PILOT LANGUAGE. The usual technical documentation is a must if any changes are to be made: the program listing, flowchart, detailed operational description, file-handling methods, modifications, extensions beyond PILOT 73, etc. It must be clear enough so a programmer not familiar with the system can make changes without rewriting the whole thing.

EDITOR DOCUMENT. See above and repeat.

PILOT TUTORIAL. As this is in PILOT, a listing and chart indicating the main branches is sufficient.

USER'S MANUAL. This is perhaps the most important document. A new user should be able to learn PILOT with the manual and a terminal. Any good manual has these three levels: (1) How to do the mechanics, such as logon and type "return." (2) How to use PILOT and the Editor. (3) Advanced things and neat tricks for a user with some experience.

THE SEED PROGRAMS GUIDE. More than an index, this document gives brief summaries of each seed program and its intent. At a later time this simple guide can grow into a comprehensive manual of techniques for teachers. An accompanying document provides the listings.

IMPORTANT PEOPLE

The people involved in your PILOT project are very important for its growth. Here are a few roles—usually different persons fill each one as the styles differ in each case:

THE PROGRAMMER. This fellow (person) gets the PILOT language, the editor and some other software running. He knows all its quirks and is the sole source of help when it's bug-fixing or upgrade time.

THE WRITER. He/she writes the manual and other documentation for users. This person does not care much for the details of the system...just how it can be used. The motivation is to give a clear, simple presentation to the user.

THE TEACHER OF TEACHERS. Here the motivation is towards teaching instructors how to use PILOT for their applications. This person MUST appreciate the biases and viewpoints of the naive beginner. Especially those about the computer—how it is impersonal, etc. Patience and a winning personality are crucial for this role.

THE COLLECTOR. This person is interested in finding and making available interesting PILOT programs, manuals, etc. He will grow a system library of PILOT programs and improve the collection of seed programs. Good taste and a librarian's sense are important here.

The last three roles are often ignored by the programmer or else the computer-center staff thinks it can do it effectively. This is rarely the case and the staff should welcome those who do become attached to these roles.

Too often good systems are represented by those who have little conception of the human needs of the users. PILOT really requires the right people for innovation and active usage. You can easily imagine the alternative.

OTHER RESOURCES

To paraphrase—good systems do not grow alone. A local user's group should meet from time to time for sharing and management of the PILOT system. Contact with other installations greatly increases scope and fertility. In summary, you need more than the processor for smooth and rapid implementation of PILOT on your system. Take heed.

ACCESS TO INFORMATION

At present there is very little in print regarding PILOT. The PILOT Information Exchange has a newsletter with indexes to implementations, users, seed programs, manuals, technical specifications and applications articles, and a library of unpublished PILOT materials. PILOT materials are available for the cost of copying. (The PILOT Information Exchange, c/o Loop Center, 8099 La Plaza, Cotatl, Calif. 45628.)

SOME VIEWPOINTS TOWARDS PILOT

OVERCOMING FEAR OF COMPUTERS

The public's image of the computer is highly negative: computers are tireless malevolent malicious beings whose intent is to build a sterile anti-human world. Humans are to be manipulated, bent, folded and spindled to the arbitrary whims of the machine-god, according to this view.

The fault, of course, lies not in the machinery but in the programming and the institutions which use computers. It is always "computer errors" or "go see the computer" rather than specific human beings.

In the LOOP center, a storefront computer center open to the public, some common reactions are: "Computers intrigue me, but I am afraid of them." "Can I ask the computer questions and get answers about anything?" To overcome these biases requires both patient and human-oriented people to tend the initial man-machine interaction between the shy user and the computer. Also vitally important is the software's capacity to respond lightly and humorously and "humanly" rather than mechanically.

Human beings usually communicate via dialogues made of words. Most computer languages are ill-equipped to handle words.

PILOT is a simple language which is entirely word and dialog-oriented. It is far easier to write a simulated dialog in PILOT than in other languages. This includes languages such as COURSEWRITER, which are designed for educational uses.

PILOT has several features which aid the creation of dialogs. It has a minimum of the syntax that confuses the word-oriented human program writer. PILOT has a powerful word-matching function which identifies likely keywords in anticipated responses. Another feature is the capacity to echo selected responses at later points in a dialog. This simple feature vastly personalizes and increases the intimacy of a dialog. (For example, the reply to "What's your name?" can appear elsewhere.)

Overcoming the public's fear of computers will take a long time. Individually responsive dialogs such as those possible with PILOT will help.

WHY KIDS SHOULD WRITE IN PILOT—A HIDDEN RESOURCE

The standard educational computer system has only two goals. One, teach the material; two, watch the students. The approach definitely limits the student's role to the traditional one of a semi-sentinent sponge.

Yet—let the student actually program the machine, and WATCH OUT!!! If it is a BASIC system, half a dozen games will suddenly sprout and get intensive use. (Has anyone noticed that these games never seem to fill any particular educational need, yet use sophisticated concepts for irrelevant purposes?? A statement of educational value.)

PILOT is especially designed for student use. PILOT's simple form is easily learned by 10-year-olds. Kids will be writing short programs in a day or two—the wise teacher can use this property to great benefit. Here is a short guide:

Traditionally, the purpose of the essay question in tests is to see if the student can express his knowledge clearly and concisely. If he can, it's a good assumption he knows not only the material but also how it is organized. This principle can obviously be extended to the creation of a clear teaching program to illustrate the points of understanding.

BE CAREFUL OF RFI AND TVI FROM YOUR CPU

The title might sound like a mouthful of buzzwords but your neighbors will be calling you other words if you're not careful of this problem. Namely, that some of the "naked" (one board, no cabinet) microcomputers and even the ones in cabinets when the cover is off put out some interference signals in many frequency ranges. The reason for this is that the digital signals are, of course, square wave pulses occurring at different cycle rates. Furthermore, a square wave can be thought of as a summation of harmonically-related sine waves, the high-order harmonics of which can cause RFI and TVI.

Several solutions suggest themselves. 1) Use your system in a cabinet whenever possible. If yours is a one board system, mount it in a metal chassis. 2) Construct a shield out of extremely fine screening (coarse screening provides little or no protection). 3) Use a large filter on the AC line.

TEN WAYS TO SPOT A COMPUTER EXPERT

by
Chuck McMichael
158 Freeport Rd.
Butler, Pa. 16001

1) Does he talk a lot about *hardware* without being able to tell the difference between a gasket and a hexagonal lug nut?

2) Does he often use the word *"batch,"* while failing to follow it with the phrase "of cookies"?

3) Does he claim to know several *languages,* none of which are taught by the Berlitz School of Home Instruction?

4) Does he often use the expression *garbage in—garbage out,* even when he's not eating at the cafeteria?

5) When playing cards, does he turn pale and shudder each time someone says "shuffle the *deck*"?

6) a) Does he *print* a lot, using capital letters?

b) Does he spell funny, taking *words* like "cat" and "square root" and making them into "KAT" and "SQRT"?

7) When asked to name his favorite *program,* does he choose RANDU instead of *Happy Days?*

8) While making Christmas wreaths out of *computer cards,* does he stop to read what they say?

9) If you tell him you've ruptured a *disk,* does he ask what was stored on it?

10) Does he say "line" or "queue"? Whereas most people say "line," an expert will say "queue"; you can take this as a cue that computers are his line.

It is easy to take advantage of this idea in PILOT. Give a few example programs which clearly teach a simple idea and set the students to writing little PILOT tutorials on whatever subjects. Some of the programs will be good and a few excellent. Let the entire class interact with these. The students will share each others programs with mutual critiques and a synergistic learning process starts. In this way, a teacher's role changes from materials spoon-feeder to that of mentor.

A subtlety of this method is the language, style and tempo of the student-created material will often communicate a subject more effectively than one written by adults. Children of different ages use language differently and this will be reflected in the programs. Also, ethnic differences can be taken into account.

You may note that this is heresy: students can be an educational resource of enormous value. PILOT is one of the few languages which can tap this resource.

THE ROLE OF FLUIDITY, MUTABILITY AND CHANGE

All too often the computer destined for education gets waylaid by the idea that "Once we have it programmed right, it will teach all the courses for us." The whole notion is that an initial large-scale investment of time and money will yield lasting long-term savings.

The catch to all this, of course, lies in the time required to get that first system completed. Inevitably it is in the range of several months to a few years, and somehow, the system never is really ready. A moment's thought will show why. First, the external requirements, such as state rules, budget changes, staff changes and so forth, change during the initial period—usually forcing corresponding changes in the computer-based curriculum under development. Also, the teachers and courseware authors are making improvements (?) to their product.

PILOT offers a different approach to this problem. Why have a large system of courses built by a few experts to fill every need? It can't be done and costs too much. There are more educational computers running BASIC than there are running large CAI systems.... As with BASIC, PILOT programs may have a very short lifetime (such as hours or days) and yet can be extremely useful in teaching.

Programs with a short lifetime encourage experimentation and novel approaches. If it doesn't work, throw it away and try something new. If it does work, improve it or save it for later. Most important, the teacher and students can make these choices without fear of a negative evaluation from higher authority. No curriculum committees and month-to month haggling is required before a new idea can be tried.

Ephemeral PILOT programs are also light in tone and mood—that is, they aren't "serious" or necessarily yoked to some vast task. Any creative student knows the importance of lightness and triviality in true teaching.

SUMMARY

I hope these remarks have given you a taste of PILOT's flavor and usage in an educational context. PILOT fills an important space in the area of word-oriented, student-authored programs. It is intended as a classroom tool for teachers not especially oriented toward computers and is not specifically aimed at curriculum-makers. PILOT's simplicity is a great aid for the computer-reluctant or time-short person. The four mnemonic instructions, TAMJ, allow effective programs to be written immediately. ■

The author, a student at MIT, says he wanted to cover the "flavor" of the language, not write a reference manual.

A Taste of APL

Craig A. Finseth*

APL. What are those three letters that keep coming up, whispered in terminal rooms, barely audible under the sound of the Teletypes? Why do these letters keep coming up, even after they have been quashed for the *n*th time? And what could be in a language that would turn an otherwise ordinary programmer into something that has forgotten what a DIM statement is and doesn't even look at programs over ten lines long?

Let's start with its history, brief as it is. In the early 1960s, Dr. Kenneth E. Iverson published a book called *A Programming Language* (Wiley). This book described a new type of mathematical notation that would be self-consistent, in that instead of the current system in which different functions obey different syntactic rules (for example, 3×4 vs. tan 56) there would be one uniform rule. Each operation has its own character and appears as operator-argument or argument-operator-argument, depending upon whether it takes one or two arguments. (Subtraction takes two arguments while absolute value only needs one.) One character can have two different meanings depending upon context. Anyway, there it sat for awhile, until IBM came into the picture. It seems that they had put a computer into a research lab and it wasn't being used. They implemented a modified version of Iverson's notation and usage went up tremendously. In the first year, over 100,000 terminal hours were logged. The language eventually became APL\360 and APL has been expanding slowly ever since.

In APL, the workspace contains all function definitions, variable values, and anything else needed in order to run your programs. All of APL's built-in functions (usually called operators) are represented by single characters; user-defined functions have names just like variables. Anything that you type in is immediately evaluated from right to left. This means that running a program is actually just calling a function.

But enough talking for now! Let's go through a terminal session and see some of the details.

We want to start with a clear workspace (sort of erasing the blackboard), so we type in this:

```
      )CLEAR
CLEAR WS
```

(Your input is automatically indented six spaces by APL in order to distinguish it from the output). The ")" just means that it is a system command. Now, to find out what 3×4 is:

```
      3×4
12
```

As was mentioned, APL evaluates anything that you type in. To store the result and not print it, enter:

```
      A←3×4
```

The variable A should have the value 12. To print the value, just enter its name.

```
      A
12
```

At this point, it is worthwhile to note that APL uses "x" for multiplication and "÷" for division; "*" signifies exponentiation and "/" will be explained later.

In APL you can assign an array of numbers to a single variable. The numbers are separated by spaces. For example:

```
      A←1 2 3 6.7 9
      A
1 2 3 6.7 9
```

assigns the vector 1,2,3,6,7,9 to A. Now, whenever we refer to A, we are simultaneously referring to all of its elements. You can also get any individual element or group of elements by indexing.

```
      A[4]
6.7
      A[1 4 5]
1 6.7 9
```

For arrays of higher dimension, the indices are separated by semicolons. Unlike BASIC or FORTRAN, arrays are completely dynamic and can change in size or number of dimensions at any time. A way of creating arrays of higher dimension will be covered later. There is also the special case of a null array—a sort of representation of nothing. Just like the invention of zero, the null array has proven very useful. Now, since we won't be using A anymore, let's get rid of it.

```
      )ERASE A
```

And it is no more. Now, we shall define a function to generate all primes less than a given number. We will use the sieve of Eratosthenes to take all numbers less than n and drop the non-primes by checking for divisibility by the integers less than the square root of n. To start with, we specify a header giving a name and other information for the function.

---

*916 Ebony Ave., Duluth, MN 55811

```
        ∇R←PRIME B
[1]
```

The triangular character, called a del, tells APL that we want to open function definition. The variable R will be assigned the result, and the variable B is the one argument. APL responds by giving us a line number, inviting us to continue. We will start by giving I the value of 1. I will be our current divisor.

```
        I←1
[2]
```

And APL prompts us with another line number. Now, we assign R the vector from 2 to B. This means that R has the value of 2,3,4,5...B. The thing that looks like a mangled 2 is called iota and means to start with 1 and generate a vector of all the integers until you get to the number on the right. The down-arrow means drop. In this case, it drops 1 element (the first) from the vector on the right. (For explanation purposes, we will look at the case where B is 9—keep in mind that B is a variable.)

```
        R←1↓ιB
[3]
```

Note that we already know that one is defined as not prime. Remember that APL executes from right to left. Also, there is no heirarchy of operators, so that operations are done in the order they are encountered, except when they are within parentheses. Thus, in the example:

```
          5×7-3÷4+5
 33.33333333
```

APL add 4 and 5, divides 3 by this sum, subtracts this from 7, and multiplies by 5. To do this in BASIC or FORTRAN, you would have to do this:

5*(7 – 3/(4 + 5)) (*not* APL notation)

You might ask: why should APL do this differently than other languages? APL has many more operators than BASIC or FORTRAN, which only have five (+, –, *, /, **). This creates the problem of deciding which to do first. (Do you do factorials before or after arcsin?) Iverson solved this problem by deciding that APL should have no heirachy.

Let's continue with our function.

```
        LOOP: R←((I=R)≥0=(I←I+1)|R)/R
[4]
```

No, don't go away, this mess has meaning. We'll start on the inside parentheses, keeping in mind that the /R is "hung," waiting for the stuff inside the parentheses to finish.

```
        (I←I+1)
```

This just increments I by 1.

```
        (I←I+1)|R
```

The vertical bar means, for positive arguments, the remainder of an integer division of the left argument into the right. Some examples:

```
        5|12
2
        3|6
0
```

It should be pointed out that if you have a single number on one side of an operator, and an array on the other, the scalar will be expanded to match the array in size. Thus, the scalar I and the vector R can be processed by the vertical bar.

In APL the six relational operators produce results based on the truth of the relation, where one stands for true and zero for false. So 3 = 4 would produce zero while 5 = 5 would produce a one. Combining this with scalar extension makes sense out of the "0 = (I...." Whenever the result of the vertical bar is not zero, the result of the equals operator is zero and whenever the result of the vertical bar is zero, the result of the equals is one. The net effect is to produce a zero whenever I divides evenly into a given element of R, otherwise a one.

We have to be sure that an element is not lost because I is equal to it. So we have the I = R set a one where I is equal to a given element of R; everything else is zero. The greater-than-or-equal-to sets every element in the result to one, except where I≠R and the vertical bar has a zero remainder. The following diagram shows the intermediate results (each starts below the operator that generates it, with the lower levels being generated later).

```
((I=R)≥0=(I←I+1)|R)
          2     2 3 4 5 6 7 8 9
                0 1 0 1 0 1 0 1
          1 0 1 0 1 0 1 0
 2  2 3 4 5 6 7 8 9            VARIABLE
   1 0 0 0 0 0 0 0              VALUES
      1 1 0 1 0 1 0 1
```

Now, after all that, we have a vector of zeros and ones—why do we want it? Well, remember the /R on the end? The whole line is:

```
[3]     LOOP: R←((I=R)≥0=(I←I+1)|R)/R
```

At last, the explanation of /. Let's say that we have 1 0 1/3 4 5. This will result in 3 5. Wherever there was a one, the element is kept and if it was zero, the element is dropped. Thus, our 1 1 0 1 0 1 0 1/2 3 4 5 6 7 8 9 results in 2 3 5 7 9—the multiples of two have been dropped completely! This result vector is assigned into R and the statement is finished. (LOOP is a statement label, a variable that contains the statement number of the line that it is on (i.e., 3).)

Now, for the last line.

```
        →LOOP IF I<B*0.5
[5]     ∇
```

This goes to the line number given by LOOP if I is less than the square root of B. Line five has a del in order to close function definition.

Let's execute the function and see what happens.

```
        ANSWER←PRIME 9
SYNTAX ERROR
PRIME[4]    →LOOP IF I<B*0.5
                         ^
```

What happened? We made a mistake. APL printed the message, the errant line, and a carat pointing to where it blew up. It turns out that we forgot to define a function IF. Keep in mind that only operators are predefined. In general, if it has a name we have to define it ourselves. So:

```
        ∇R←A IF B
[1]     R←B/A
[2]     ∇
```

In this function, R will be assigned the value of A if B is true (one) or else null (here's one of its many uses). It can get the null value because that's what's left if you remove everything from A via the "/" by having a false (zero) value for B.

APL keeps each function separate, so all line numberings begin with 1,2,3... Now, we can pick up where we left off by doing:

```
→4
```

in which APL goes to line 4 of the most recently suspended function. (Most often, functions are suspended by errors.) Since there is only one suspended function, APL goes to line 4 of PRIME.

Now, it just prints a linefeed. To find the answer enter:

```
      ANSWER
2 3 5 7
```

One of the nice things about APL is that if you wish to do something that is not a defined operator, or just do something in a different way, you can define a function to do it. Also, by writing the proper functions, you can readily model almost any other language in APL.

One final note about the right-arrow. It causes APL to go to the indicated line number. If the line is not there (say, line zero) the function will stop. If it is a null value, APL just goes on to the next line.

We mentioned that APL can handle arrays of any dimension, but was not said how to create them. To generate a 2-by-3 array of zeros you do this:

```
A_MATRIX←2 3ρ0 0 0 0 0 0
        or
A_MATRIX←2 3ρ0
```

(Remember scalar extention). Now the variable A MATRIX has a shape of 2 × 3). This means that it has two rows and three columns.

```
      A_MATRIX

0 0 0
0 0 0
      ρA_MATRIX
2 3
```

Using just one argument with that funny-looking ρ (called rho) gives the shape (the length of each dimension) as a result.

Suppose that you want to find the location of a value in a vector. For variety, we'll use characters, which are treated just like numbers and stored one character per element. Anyhow, we want to use the two-argument form of iota.

```
      'ABCD'ι'C'
3
```

'C' is, of course, the third element in 'ABCD'

APL can also meld two items into one. For this, the concatenate operator (comma) is used.

```
      FIRST←2 3ρ'ABCDEF'
      SECOND←2 4ρ'GHIJKLMN'
      FIRST

ABC
DEF
      SECOND

GHIJ
KLMN
      BOTH←FIRST,SECOND
      BOTH

ABCGHIJ
DEFKLMN
      ρBOTH
2 7
```

APL can also transpose a matrix (switch its dimensions)

```
      ARRAY←2 3ρι6
      ARRAY

1 2 3
4 5 6
      ρARRAY
2 3
      TRANS←⍉ARRAY
      TRANS

1 4
2 5
3 6
      ρTRANS
3 2
```

And, to reverse an array (flip it right-to-left) do:

```
      REV←⌽ARRAY
      REV

3 2 1
6 5 4
      ρREV
2 3
```

You can also rotate an array any amount.

```
      ROT←1⌽ARRAY
      ROT

2 3 1
5 6 4
```

Here, ARRAY has been rotated one element to the left. It should be noted that APL allows names of almost any length. They have been kept short for convenience.

There are a number of functions that will just be mentioned before we move on. APL supports the logarithm and exponential functions, trig functions (sine, cosine, and tangent) and their inverses, the hyperbolic functions, pseudo-random number generation, sorting, the logical functions (and, or etc.), matrix division, generalized inner and outer product, generalized summation and generalized cumulative sum. Generalized means that any function that takes two arguments can be used instead of just plus or times.

Two more things before we move on to input and output. Negation (−) (the one-argument form) means to take what is on the right and change its sign. Negative (¯) means that a number is negative. Thus, −3+4 means ¯7 while ¯3 + 4 means 1. Also, if A is a variable then – A is legal while ¯A is not.

APL has two operators that handle input and output. They are:

```
QUAD:  ⎕        AND     QUOTE-QUAD:  ⍞
```

For output, just assign into quad.

```
        A←3
        ⎕←A
3
        A
3
```

For input, just assign from quad.

```
        A←⎕
⎕:
```

APL then waits for you to enter any expression. This will be evaluated and assigned into A. So we type:

```
        7-10
        A
¯3
```

This can cause problems, especially for instructors who are writing drill-and-practice programs. When they type "WHAT IS 11 × 5?" as the answer. Instead, they can use quote-quad for input. Quote-quad just takes the input and stores it as characters. A sample line might look like this:

```
VALUE←⍎(INPUT∊'0123456789')/INPUT←⍞
```

First, the input characters are stored in INPUT. Then, that fishy-looking e (called epsilon) produces a one whenever a character on the left is in the array on the right. Thus, only where the student entered a digit will there be a one. The slash then deletes everything but digit characters. That odd-looking thing, just to the left of the "(", takes the character vector on the right and treats it like an APL expression. This VALUE has the numeric value of the character vector, which itself has had everything but valid numeric characters deleted.

One final example, and I'll leave you to your hundred-line BASIC programs (you should be able to see by now that a little APL can do a lot).

Let's define a function CHANGE which will print the number of half-dollars, quarters, dimes, nickels, and pennies to be returned from a purchase.

```
        ∇CHANGE
[1]     0 2 2.5 2 5⊤100-⎕
[2]     ∇
```

This function doesn't return a value or require an argument, so all that's on the header line is its name. The one and only line subtracts the input (via the quad) from 100 and then proceeds to apply the represent operator (it looks like a t) to that value. Represent multiplies all the elements together and sees how many times that goes in, then places this in the first element of the result. It repeats this with the all-but-first elements, then the all-but-first-and-second, etc., storing the quotients in successive elements of the result. Let's run an example.

```
        CHANGE
⎕:
        17
1 1 0 1 3
```

This says that there is one half-dollar, one quarter, no dimes, one nickel, and three pennies in 83¢.

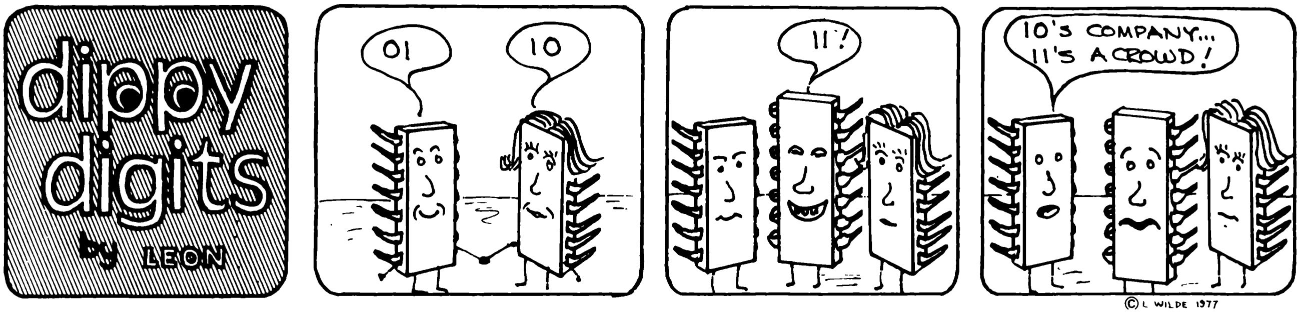
Let's see which functions we have defined and list them.

```
        )FNS
CHANGE   IF          PRIME

        ∇CHANGE[⎕]∇
      ∇ CHANGE
[1]     0 2 2.5 2 5⊤100-⎕
      ∇


        ∇PRIME[⎕]∇
      ∇ R←PRIME B
[1]     I←1
[2]     R←1↓⍳B
[3]   LOOP: R←((I=R)≥0=(I←I+1)|R)/R
[4]     →LOOP IF I<B*0.5
      ∇


        ∇IF[⎕]∇
      ∇ R←A IF B
[1]     R←B/A
      ∇
```

In summary, APL is a concise, powerful language with applications in many areas. It is built upon arrays and user-defined functions, thus it is amazingly flexible and versatile.

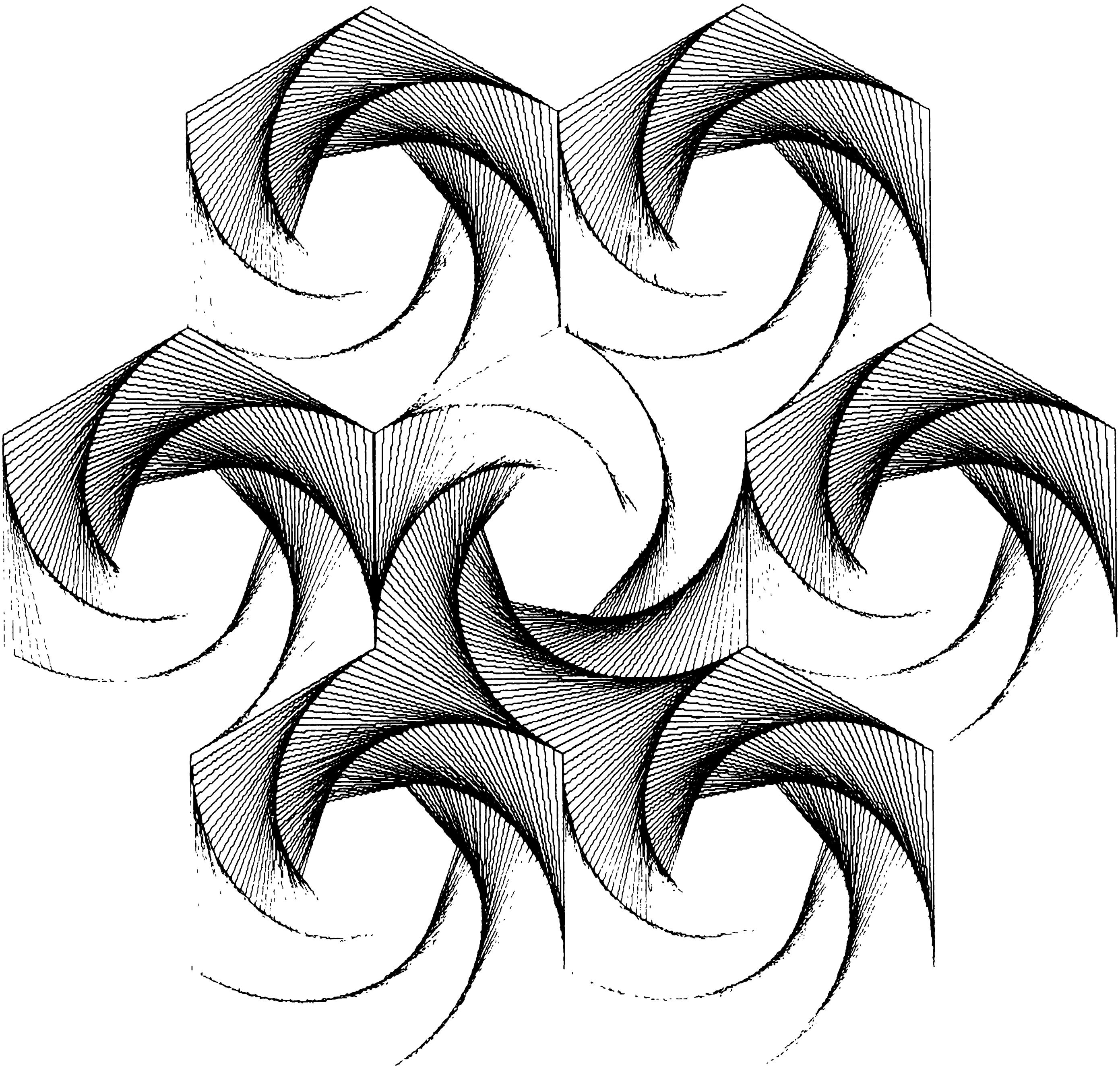
This is only intended to give the "flavor" of APL. The best way to learn the language is to get a manual and sit at a terminal, experimenting.

Funds for examples provided by the Student Information Processing Board of the Massachusetts Institute of Technology. ■



ARTSPEAK — A Computer Language For Young At Heart And The Art Lover

Jehosua Friedmann*



*5 Molcho St. Rehavia
Jerusalem, Israel

In the following we intend to discuss the ARTSPEAK language, its areas of applications, the required software and hardware is then briefly outlined, followed by a programming example along with samples of output. This article is in no way a tutorial in ARTSPEAK.

What Is ARTSPEAK?

ARTSPEAK is a specialized computer language which provides graphic output. That is, the language has the potential and capability, when used properly, to output graphs, designs, curves of all kinds and an unlimited variety of original artistic drawings, some of them with stunning effects. This computer language is unique in that it has a limited syntax (grammar) and its statements are simple self documenting English statements. The language can be taught to adults and children. Artspeak is easy to learn and it can be mastered in hours.

Short History

In the last decade there was a tremendous increase in number of students taking computer courses, In particular, at present, there is a strong demand for a computer course on an exposure level like the course titled Computers and Society given in many colleges. The motivating force responsible for the generated interest was the increasing and all pervading influence of the digital computer in almost all facets of our life. At the time a child is born he becomes a number in the computer. Later, in school and in college, he uses the computer for registration, statistics, tuition payments, and even for some of his education. Upon entering adult life, his salary, federal state and city taxes, his savings and checking accounts are all controlled by a computer. When he takes a vacation, or he makes an airline reservation or perhaps rents a car he is hooked to a computer. Even when he enters a hospital he still can not escape the influence of a digital computer. In short, the computer found its way into business, military, government, educational institutions, industry and all other areas of live endeavor. Our society is by now so committed and dependent on the computer had it stopped, our national economy would cease to function.

It is generally believed that to learn about computers one must interact with them. Interaction means communicating with a computer through a computer language. However, programming (writing programs using instructions which tell a computer to perform a specific task) requires a logical approach and mind, an ability to solve problems and at least an appreciation, knowledge and in most cases an ability to tackle mathematical problems.

It is an open secret, that a large number of our high school and college student population inherited a fear towards any subject that has a smell of mathematics to it. As a consequence, teaching to students a computer language became a difficult and often an impossible job.

This problem was recognized by Dr. J. T. Schwartz of the department of Computer Science at New York University. The story so goes, at a lunch one day he outlined on the back of a table napkin the elements of a new computer language which he called ARTSPEAK. The objective was to design a language which was very simple, easy to learn, easy to write and understand, with a minimum instructions and the output was to be something pleasing to the eye.

In 1970, Mr. David Benevy, then a graduate student, (and now at Tel Aviv University) wrote a compiler for ARTSPEAK to be used on a CDC 6600. Shortly thereafter the language was improved by several students at New York University. Currently the compiler is being modified by Dr. Caroline Wardle from the Mathematics department Hunter College, New York, for use on an IBM 360/370 computer. The Artspeak compiler may be obtained by writing to the Service Bureau of Control Data Corporation, New York City. A manual describing ARTSPEAK, titled The Art Of Programming ARTSPEAK by Henry Mullish, may be obtained from the Courant Institute of Mathematical Sciences, New York University, Washington Square, New York City, N.Y.

Reaction of Users

Students using ARTSPEAK found that they could write programs within an hour of instruction. These programs required no artistic skill or great intelligence. The graphical output generated a very positive attitude toward the computer as well as an appreciation of its potentialities. As a consequence, many students set upon improving and enhancing their programs to an extent that some masterpieces were produced, as a result.

Even children, at two experiments performed at New York University, learned very quickly to write computer programs using ARTSPEAK. Some of these children produced very interesting graphical designs and pieces of art. Most of the children said that they thoroughly enjoyed their experiences with ARTSPEAK.

A Programming Example

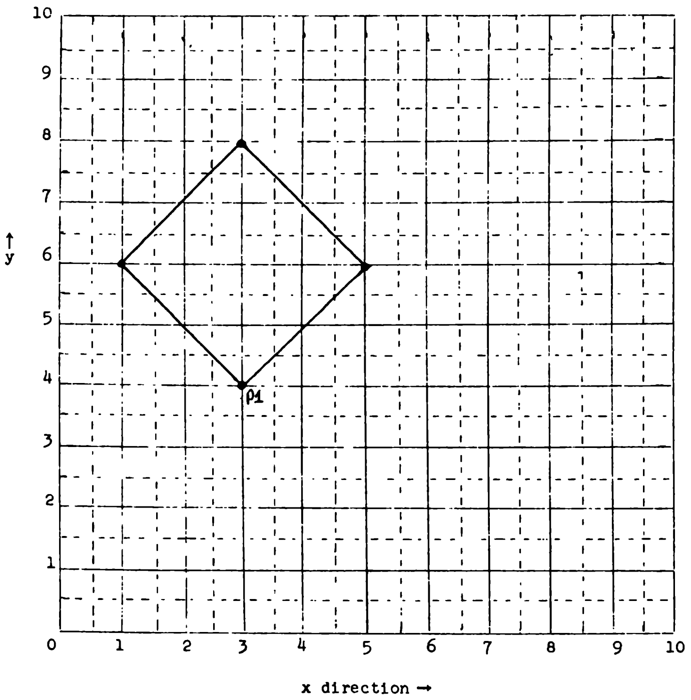
ARTSPEAK is not an interactive computer language. Generally, writing a program involves punching each instruction on a separate card (the language provides continuation of an instruction to the next card if it is too long.) The general format is similar to the one found in FORTRAN. After the cards had been punched they are assembled along with some control cards for submission to the computer center. To execute a program in ARTSPEAK the computer center must use a CDC 6600, an ARTSPEAK compiler, a card reader, disk, console terminal, printer and a plotter. After the program had been executed it is returned to the programmer along with a 10"x10" output sheet. The output includes a program listing and some output (if the ARTSPEAK compiler found no violation of the language). The programmer, then examines his output if he is not satisfied or he wants to improve his program or experiment with it, he might make changes. Usually, these changes would involve addition or deletion of one or more statements. These statements are punched on a keypunch machine and the program is resubmitted for execution.

All drawings in ARTSPEAK, that is, the output, use paper 10"x10" in size. A programming aid called Artspeak Design Blank, shown later with a sample problem, was developed to facilitate designs. This form has 10 units on the X and 10 units on the Y axis. Thus, a total of 100 squares are available for planning purposes. Each of these squares corresponds to the one inch square area on the output form. To initiate a program, one first sketches his design on the Artspeak Design Blank. Then, using this design as a guide the programmer writes his instructions, which thereafter are keypunched to form a program.

ARTSPEAK Design Blank

Programmer's name: JEHOSUA FRIEDMANN

Design name or #: SQUARE IN MOTION



In the following we will illustrate, plan, code and explain a sample problem using ARTSPEAK.

Draw a square standing on one of its corners. Rotate the square on its corner an angle of 2° degrees and redraw the square. Repeat this process 45 times and finally stop. The ARTSPEAK program was written in two stages.

- We planned the program.

We have used the artspeak design blank form to do the planning. First, we have chosen a point with the coordinates (3,4) and marked a dot on its place. This point was labelled P1. We choose three more points (3,8), (1,8), and (3,4) again using a dot to mark their position on the form. Finally, we connected these points with lines which formed a square standing on one of its corners.

- The artspeak program-coded.

```
1           LET P1 BE POINT (3.4)
2           LET C1 BE LINE P1, (5,6), (3,8), (1,6),
            (3.4)
3     L1    DRAW C1
4     L2    ROTATE C1 ABOUT P1, ANGLE 2
5           REPEAT L1 TO L2, 45 TIMES
6           STOP
```

Explanation of the program.

Statement 1 tells the computer to define a point at 3,4. This point is labelled in the program as P1 as was also done on the design form. (In artspeak a point may be represented by using the letter P followed by any positive integer selected from 1 to 100)

Statement 2 then defines a curve which connects these points to form a square. This square is formed starting at (3,4)=P1 and connecting the points (5,6), (3,8), (1,6) and finally joining with point (3,4). The square identifies a curve labelled C1. (In artspeak curves are identified by the letter C followed by any positive integer selected from 1 to 100)

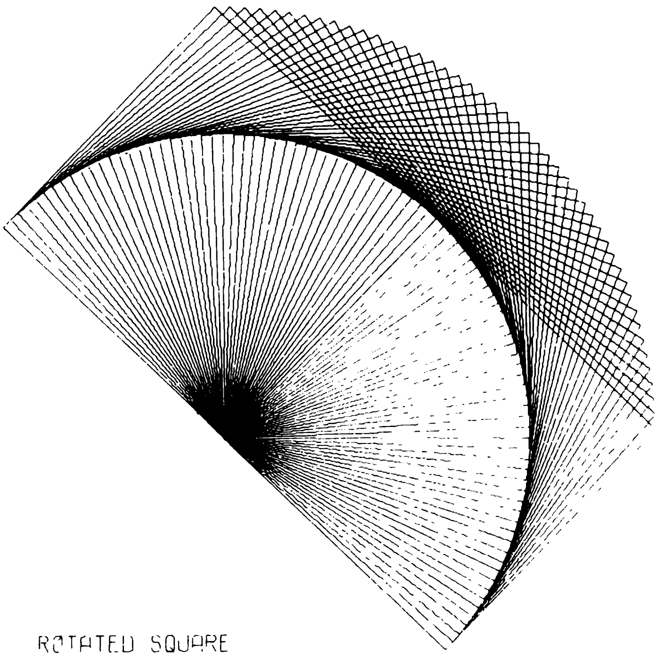
Statement 3 commands the plotter to draw the square.

Statement 4 tells the computer to rotate the square C1 about point P1 2 degrees clockwise. (You may note that artspeak allows one to assign a label to each of its statements, however, only statements which are referenced are usually labelled. All labels must start with the letter L followed by an integer from 1 to 100. In this program we have assigned labels L1 and L2 to statements 3 and 4 respectively.)

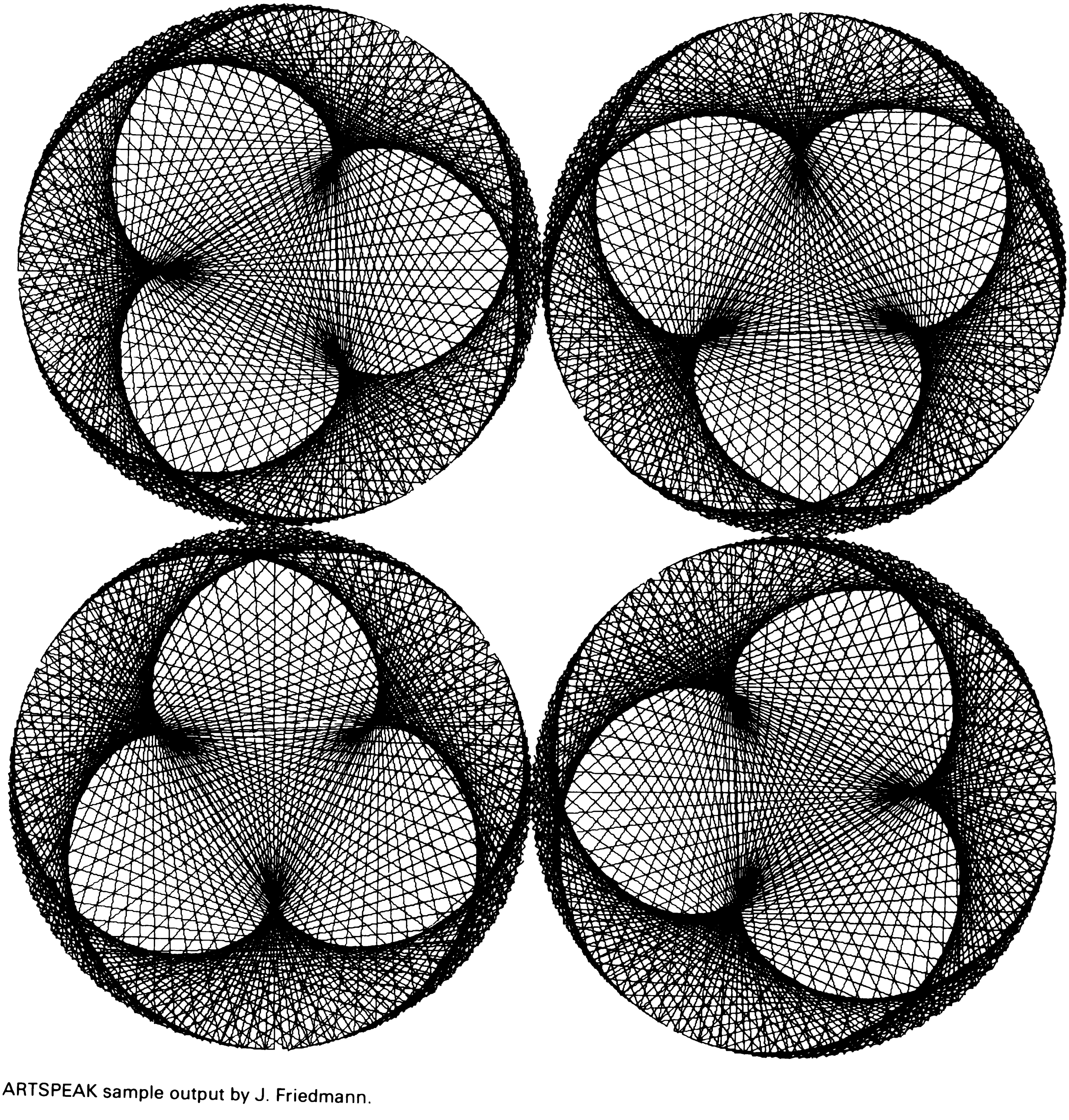
Statement 5 tells the computer to repeat this process (steps 3 and 4 in the program) 45 times.

Statement 6 will stop execution after 45 repetitions of statement 5 had been completed. ■

The output



ROTATED SQUARE



ARTSPEAK sample output by J. Friedmann.

A New Generation of Biomedical Instruments

John M. Brus

The development of four microprocessor-controlled medical devices at Biomedical Engineering Center for Clinical Instrumentation (BECCI), based at Cambridge, Mass., is pioneering the research trail to a new generation of these devices, according to the consensus opinion of the BECCI research engineers.

Jointly sponsored by the Harvard/MIT Program in Health Sciences and Technology but funded by a National Institutes of Health three-year contract, BECCI's aim is to build a technological resource offering an integrated and modularized set of hardware and software specifically designed for biomedical applications.

To this end, BECCI published last year an upper-level language called STOIC (authored by John Sacks) and developed an "on-line debugging" card, designed by electrical engineer Paul Schulter.

Staff engineer John Volvano explains that "Given the time and effort spent on the background hardware and the STOIC software base...future projects will have little of this work. Engineers need only design one or two hardware cards to interface their particular project and develop the software on STOIC"—considerably telescoping the time needed to transform an instrumentation idea into a prototype.

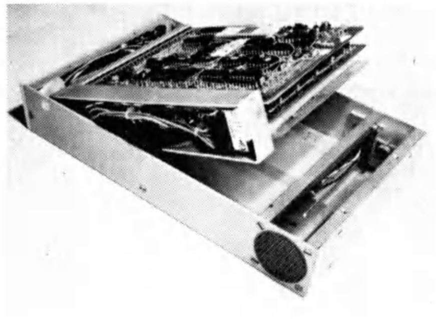
John Sacks, BECCI's software specialist, says STOIC gives the programmer complete control over the execution speed vs. ease-of-programming tradeoffs inherent between machine and higher-level languages. Additionally, the debugging card, designed for microprocessors using an Intel 8080, uses a "memory mapping" feature to conveniently "patch" programs as if they were read/write locations, avoiding frequent read-only memory reprogramming.

The purpose of the medical instruments under development is not only to monitor and analyze biological signals but also to present this information in a usable form to the physician. In another sense, however, the research engineers are confronting the question: "How do you program a physician's clinical judgement into a computer?" The best example illustrating some of these difficulties is BECCI's portable arrhythmia monitor, scheduled for limited field-testing this summer.

Monitoring Chaotic Heart Beats

There are almost 700,000 heart-attack victims each year in the United States. Heart attacks usually occur when blood flow to a portion of the heart is reduced or blocked, disturbing the natural rythmic wave of electrical impulses regulating the heart's beating. This leads to ventricular fibrillation—uncoordinated beating of the heart's chambers—and cardiac arrest. People with heart disease are prone to intermittent arrhythmias, and monitoring these patterns provides valuable medical information of the heart's physical situation and response to medication. Since the heart beats about 100,000 times in a 24-hour period, however, a 12 or 24-hour electrocardiogram (ECG) monitor generates mountains of data but only a few of the medically important arrhythmia periods.

BECCI is developing a portable, microprocessor-controlled ECG monitor that recognizes and stores only arrhythmia patterns. In theory, after strapping on the unit in the morning, a patient can go about all his daily activities. At the end of the day, he plugs his monitor into a modem and feeds the data to a hospital computer which will print out hard copy for examination.



The Portable Cardiac Arrhythmia Monitor contains an Intel 8080 processor, 256 8-bit words of read-only memory, 4096 8-bit words of read/write memory, ECG amplifier, 10-bit analog-to-digital converter, bit-serial transmitter-receiver, patient interaction interface, and a DC-to-DC power converter.

The arrhythmia monitor is an ideal illustration of a heretofore impractical biomedical device, according to project engineer Joe Walters, Jr., "This microprocessor technology is clearly opening a new area because there's no minicomputer around capable of being reduced to a box this size (2" x 6" x 10") and this power (3.5 watts)."

However, recognizing arrhythmias is just one of the microprocessor's tasks. A clock enables recording of the time intervals between arrhythmias. Algorithms also classify different types of arrhythmias and compress data for storage.

Although ECG waveforms are easily susceptible to mathematical analysis, Walters admits he hasn't quite yet mastered the trick of converting "clinical judgement" into an acceptable algorithm. Constitutional biological differences between patients illustrates one of the vexing variables. That is, what looks like arrhythmias in one patient may be close to normal in another. compounding that fact with the different possible types of arrythmias means that the arrhythmia algorithm must not only analyze different waveforms, but additionally recognize the arrhythmia as significant in context with the normal heartbeats. Also, extraneous biological impulses feeding into the ECG sensors create problems, such as triggering the monitor to mistakenly store the impulses in memory. One solution, Walters notes, is beefing up the current 4K memory to 16K or even higher. This would allow the use of STOIC and provide the extra capacity to store the extraneous impulses.

Walters also has plans of making the unit interactive with the patient. For instance, after the unite detects an arrhythmia, a buzzer can alert the patient to answer a series of preprogrammed questions presented on a small LCD display. Giving the patient a choice of "Yes," "No" or "I don't know" buttons to push, typical questions might be: "Are you dizzy?" or "Are you having angina pains?" Correlating these answers with the different types of arrhythmias creates more medically useful information.

Computerizing Pulmonary Function Testing

Another unit under development involves a combination whole body plethysmograph (lung volume capacity) and respiratory gas analysis system. Project engineer Niel Dowling says previous plethysmographs, if they've been computerized at all, usually shared time on a minicomputer. "We want to have a processor dedicated to the task to produce a cheaper, more compact system," he explains.

Effective measurement of a pulmonary system includes measuring lung capacity, lung elasticity and CO_2 and O_2 transfer efficiency, as well as how these factors change within the time of an exercise period. Dowling says the microprocessor using analytic equations (e.g., Boyle's Law) can compute lung capacity straightforwardly. Or by comparing two sets of figures (e.g., lung gas ratios and blood gas ratios), the microprocessor derives medically important information on the gas-transfer efficiency of the lungs. Currently, integrated respiratory gas analysis systems and plethysmographs are not mass produced and can cost up to $50,000 counting minicomputer support, according to Dowling. but he believes a dedicated microprocessor system could cut the current cost in half. Field testing of a prototype should begin in Boston's Peter Bent Brigham Hospital this summer.

Measuring Eye Movements To Detect Balance Disorders

Disorders of the inner ear leading to dizziness or loss of balance sometimes are diagnosed indirectly. One procedure, called electronystagmography, positions a patient on a motorized chair that rotates and tilts. Concurrently, eye movements are monitored and analyzed, since inner ear balance disorders reveal themselves in eye velocities by an integrated response called the ocular-vestibular reflex.

Project engineer John Tole says electrodes placed at the eye corners can measure differences in electrical potential caused by eye movement. Again, these eye movements are susceptible to the type of mathematical analysis microprocessors happily perform. But, just as with the ECG monitor, Tole explains that patients with abnormal characteristics (e.g., unusual head thickness or weaker neurological responses caused by extreme age) can test the flexibility of the software. So far, the algorithms are standardized for a typical middle-age patient with normal neurological response. Additionally, the microprocessor sequences all chair movements and records the chair and eye movements on a time scale because some of the reflexes seem to have a delayed reaction effect.

Measuring Blood Flow

Developing a thermal probe that directly measures blood flow in living tissues is BECCI's fourth project. By placing an electrically heated needle-like probe within the tissue or organ, sensitive measurements of the rate at which the tissue absorbs heat from the probe are made. Comparing these "heat-sink" values, in the presence of blood flow, with standardized values of tissue conduction, in the absence of flow, leads to calculations of the blood flow which carries heat away from the probe.

Project engineer John Volvano says the instrument is due for field testing this summer at the Walter Reed Army Institute of Research near Washington D.C. Aside from an easily correctable but unforseen problem of shielding the sensitive electronic components, Volvano believes the software and hardware is near perfection.

Dr. H. Frederick Bowman, director of BECCI's thermal probe project, says the probe has a number of potential uses. One may be the post-operative monitoring of surgical patients and another the monitoring of transplanted organs to assess disorders steming from restricted blood flow. Knowledge of low flow rates (known as "shock") are important in patient care.

Bowman also explained that changing the instrument's software enables the probe to monitor the concentration of other fluids. This can be important in the emerging field of cryopreservation—where donor organs are stored for future transplantation. Using a cryopreservative "biological antifreeze" to prevent tissue destruction, the probe could monitor the freezing and thawing rates and "antifreeze" concentration levels for each organ system—all of which have to be meticulously recorded to discover the optimum rate.

Cheaper but powerful computer components are obviously finding a home in biomedical instrumentation. And with a little imagination, the hand-held "body-function analyzer" used by *Star Trek's* Dr. McCoy may not seem to be so impossible after all. ■

The Miraculous Medical Microprocessor: A Look Into the Future

Pamela Weintraub

Can a mild-mannered scientist travel to Mars and back, cleverly avoiding the notorious space sickness that knocks out half of our brawny astronauts? Can a 50-year-old veteran who's lost a leg in the war run the hundred-yeard dash and win? And can dying cancer patients go into deep-freeze, confident that they'll thaw out centuries later to receive the cure? Well, not yet. But if medical microprocessors live up to expectation, they may change these unrealistic scenes into everyday fact.

From electrocardiograms with whistles and bells, to zero-gravity flight-simulators, almost any instrument can

Prof. Roger Mark (right), director of BECCI, and project engineer Joe Walters, Jr. stand next to their heartbeat monitor.

be plugged into microprocessor systems. And since BECCI engineers design microprocessors with interchangeable parts, they can asemble almost 90 percent of the hardware for any given instrument from off-the-shelf parts. Thus, as the four current BECCI projects near completion, scientists are faced with the arduous task of deciding what device they'd like to work onnext.

"We want to provide the medical community with resources to build microprocessor-based instruments," says Prof. Roger Mark, BECCI's director. "If hospitals need a special-purpose microprocessor, they can come here and we'll have the necessary hardware, software and staff to build the device."

The Computerized Body

Prof. Mark speculates that portable, battery-operated processors—such as his heartbeat monitor—will result in vastly improved artificial limbs. Scientists might develop a mechanical knee that operates as smoothly and quickly as the real thing.

Our limbs move in response to electric signals coming from the central nervous sytem and brain, Dr. Mark explains. Such signals travel through the body even if corresponding limbs are no longer intact. Thus, if surgeons attach nerve endings directly to microprocessor terminals within an artificial knee, the computer will read the body's signals and instruct the mechanical part to move just as the real knee would have.

Lightweight processors may also bring sound to the deaf.

"It turns out that the hearing aids we now have are just amplifiers," says Prof. Mark. "And just amplifying sound is worthless when you're trying to make totally deaf people hear; such people are deaf because they've lost the ability to pick up certain frequencies. So, if you amplify everything you'll turn an unintelligible squeek into an unintelligible roar. But computerized hearing aids could translate normal speech into frequencies easily understood by the deaf."

Indeed, microprocessors that improve upon the human ear may one day sharpen our other senses as well. And like the mythical Martian, we'll hook up to computerized antennae that cue us in on coming earthquakes, winning horses and the next day's weather.

Human Storage: The Deep-Freeze

While Dr. Mark feels that microprocessors will build our bodies into stronger, more durable shells, Prof. H. Frederick Bowman predicts that these computers will also unravel some anatomical mystries within. In fact, Dr. Bowman's thermal-probe project uses processors to gather information that might help scientists freeze an entire person for hundreds of years.

Fred Bowman became fascinated, early in his career, with the problem of removing heat from living material. He realized that if scientists could freeze human organs, they could also establish a biological storehouse to provide the ill with healthy kidneys, livers, hearts and lungs. The key to this wealth of human parts lay in "biological anti-freeze"—a substance almost impossible to produce without help from a microprocessor.

"Each cell will survive the freezing/thawing process provided that it's frozen and thawed within a given time-span," Prof. Bowman explains. "If it's frozen faster or slower than that the cell will die." And since organs are made up of many different cell types, freezing a heart or lung has always been virtually impossible.

"You would have had to take the organ apart, freeze it cell by cell and put it back together again," comments Prof. Bowman. "Then you'd be playing God."

But now, with microprocessors to analyze the freezing/thawing process of each cell type, scientists may soon produce a major substance to broaden the overall time-span. The biological anti-freeze could provide us with a time machine more potent than anything imagined by Jules Verne. And a society where modern-day explorers travel icily into the future looms amazingly close.

Space Bound

Mark and Bowman contemplate the promise of lightweight computers here on earth, but another BECCI scientist works toward the day when he'll use processors to aid his experiments in outer space. If Prof. Lawrence Young succeeds, he may be the first researcher to feel more comfortable in a space capsule than a Cadillac.

Rapidly spinning chairs, slick model airplanes and kaleidoscopic patterns create the carnival-like atmosphere of Dr. Young's balance-testing lab. From this unusual arena, the innovative Professor of astronautics uses processors to study motion-sickness on earth.

Earth or space style, motion-sickness results most frequently under just such abnormal, man-made conditions. And whether plagued individuals spin violently in Prof. Young's laboratory chair or travel through the Milky Way at thousands of miles per hour, the symptoms include nausea, vomiting, dizziness and appetite-loss.

Thus, while patients coming to this uniue terra-firma workshop jolt back and forth in flight-simulators or stare at whirling designs, Prof. Young thinks ahead to the day when scientists—and even ordinary tourists—will travel through space without experiencing unpleasant zero-gravity side-effects.

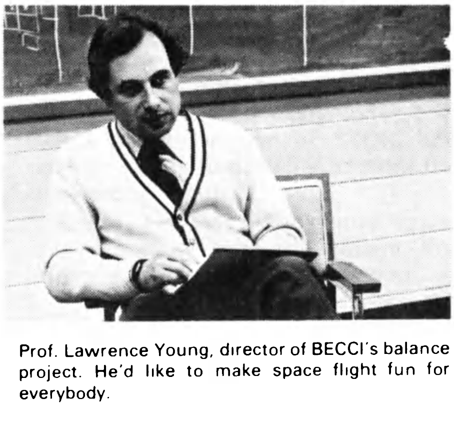
To this end he's planning a series of experiments for the 1980 Spacelab flight. In one test, astronauts ride the "space-sled"—an accelerating platform within the space ship—as scientists use electrodes to measure the rapid, involuntary eyeball oscillation associated with motion-sickness; results will be analysed via the microprocessor.

Young's work could improve the quality of outer-space living for geerations to come, with flights smooth enough to accomodate even the queasiest researcher or tourist. Passengers will learn to perform exercises that prepare them for the rigors of universal travel, and to avoid situations that produce the dreaded space disease. And since awkward design contributes to space sickness, Young's results might even create a demand for interior decorators specializing in extra-terrestrial dwellings.

And as a bonus for those who plan to stay on earth, the experiments could yield powerful cures for carsickness, seasickness and airsickness.

"The motivation for using microprocessors in the project is that they're small, cheap and powerful," says Prof. Chalres Oman, who's worked on several projects with Dr. Young. "They allow us to put a great deal of computing power into a small area—like a bedside, a clinic or a space capsule."

The microprocessor developed for clinical use is so convenient, says Dr. Oman, that scientists have even considered taking it over to the Johnson Space Center and plugging the data that comes down from orbit right into it. ■



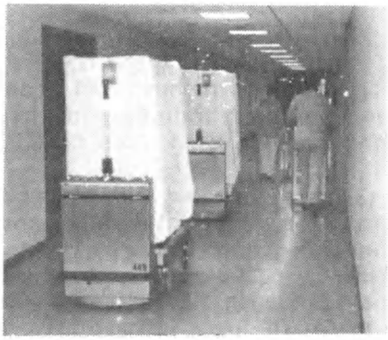
Prof. Lawrence Young, director of BECCI's balance project. He'd like to make space flight fun for everybody.

Computerized Robots: A Step Into The Future for Hospitals

Susan Trout Armstrong

A Wichita, Kansas, hospital has overcome some of its logistics problem through the use of computerized "robots."

Visitors to St. Joseph Medical Center may be surprised to see the efficient, automated carts toting linens down a corridor toward the laundry room—through tunnels and up elevators—without a driver or any visible means of propulsion.


Amscars, with modules loaded with laundry, travel the 325-foot tunnel that connects east and west campuses of St Joseph Medical Center, Wichita, KS

The helpful robots are called Amscars, and they are designed to handle a variety of hospital chores, stopping short of actual patient care. In fact, the electronically-guided vehicles transport tons of food, linens, ward supplies, pharmaceuticals and trash throughout the health-care complex.

Faced with a wide-spread hospital facility and ever-increasing costs, St. Joseph Medical Center turned to automation for help. The Amscars were developed by Amsco Systems Company, Erie, PA, as one answer to the serious problem of rising hospital costs. The St. Joseph Amscar system was the first in a Kansas hospital, and the 16th installation in a U.S. hospital.

For St. Joseph Medical Center, the robots were a welcomed step into the space-age. The sprawling complex is full of ramps and walkways that consume employee time and energy. At last the hospital could eliminate some of the non-productive time spent by personnel travelling up and down long hallways.

"Hospitals cannot automate patient care, so we are using automation where patient care is not involved," explains Mother Mary Anne, executive director of the facility.

The robots are actually a "material distribution" system, with the guidepaths in service corridors, instead of public corridors. "Because many materials must be kept sterile, they cannot be subjected to the environment of public corridors," one hospital official said. "Conversely, public corridors should not be subjected to contaminated material."

The innovative system uses Amscars to serve both the medical center's west campus and the east campus. The buildings of the two compuses are connected by a 325-foot concrete underground tunnel. The system uses clean recovery areas and soiled sending areas to prevent cross-contamination at the user levels.

Amsco Systems Company explains that the driverless carts follow a predetermined electronic guidepath which consists of special wiring embedded in the floor slabs of service corridors and covered with normal non-conductive floor-covering materials. Wires are arranged throughout the complex in continuous-loop circuits, and are energized by 6.5-KHz and 10-KHz oscillators so they radiate a magnetic field over a very short distance.

Amscar units have special coils to sense the magnetic field along the guidepath. The magnetic field is amplified to control the Amscar's steering mechanism, and the circuitry holds the robot to within a fraction of an inch of guidepath.

Transmitters, receivers, electric eyes and other devices enable the Amscars to stop and wait their turn at intersections, open and close automatic security doors, and enter and leave vertical lifts (elevators) at the correct floors, all automatically.

The robots carry various enclosed containers called "modules" throughout the medical center on pre-selected routes. For example, one Amsacar may transport a locked module full of pharmaceuticals from the supply area to the patient ward. Another Amscar might carry an insulated food module—designed to keep meals hot—from the kitchen to the nurse's station.

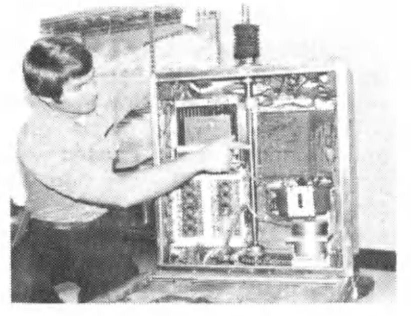
At St. Joseph Medical Center, the Amscars operate on a 16-hour-per-day schedule. Staff members at the dispatch station load the cars and automatically send them on their way. At the receiving end, other personnel unload the robots and press a start button to return the unit to its proper station.

At the Control Station, a dispatcher can monitor the Amscar lifts. Essentially, he uses a desk-type unit with an indicating panel (one for each Amscar lift) mounted to its top surface. The panel contains lights which indicates:

1. Location of the lift (floor).
2. Status of the lift (vacant or containing an Amscar).
3. Amscar at lift entrance position (at each floor).
4. Amscar at lift exit position (at each floor).

Other lights on the panel indicate possible malfunctions, such as "door obstructed."

Safety devices are included within the robots, too. They are responsible for halting the cars when they come into contact with people or other obstructions. The units travel at one mile an hour during automatic operation, but the motor provides for dynamic braking as well as for propelling the vehicle. One Amscar safety feature is the pneumatic bumper assembly. Air pressure created by bumper impact activates the bumper switch, and the car stops automatically. Still, employees say they steer clear of the vehicles as the Amscars go about their work, steadily travelling the halls with hot trays of food or dirty dishes, clean or soiled linens, or other hospital supplies. ■


The manager of the Amscar System at the Medical Center shows the complicated circuitry within the Amscar itself

Computer Correction of Optical Illusions

David A. Smith
Department of Mathematics
Duke University
Durham, NC 27706
and
Case Western Reserve University

A computer-driven plotter is a very handy tool for drawing optical illusions, especially those that depend on distortion patterns consisting of many curves or that involve complicated curves that must be drawn accurately. In a recent study (3) we have shown that the same computer plotter, given a little help from mathematics, can also "correct" illusions, in the sense of causing to appear what was really there in the first place, but didn't seem to be. As we shall see, a corrected illusion is really another type of illusion. Each of the accompanying illustrations shows an illusion in part (a) and a corresponding corrected illusion in part (b). We will discuss them individually below, but in part (a) all the lines are straight and the curves are drawn without fudging, whereas in part (b) some very specific fudging has been provided by the computer program.

It should be noted that this article is not about mathematical recreations. Understanding how and why our eyes play tricks on us is an important step in understanding the structure and function of the human visual system and its various parts. This is an area of active research by physiologists and psychologists of perception, and many important questions remain unanswered. An important preliminary step is to describe accurately what we perceive and how it differs from what is really there. One measure of accuracy of a descriptive model for perception of illusions is its ability to reverse the perceived effect and produce a corrected illusion that appears to show what did not appear when the figure was drawn "correctly."

The underlying principle on which the descriptive mathematical model is based was first hypothesized by F. Brentano about 80 years ago: *The human visual system tends to overestimate acute angles and underestimate obtuse angles*. This is certainly not an immutable law of nature, but there is a great deal of evidence to support its validity, at least among adults in civilized societies, living in a largely "rectangular" environment. The Brentano Hypothesis may be formulated precisely and applied to plane optical illusions by means of two-dimensional vector calculus. This leads to algebraic or differential equations whose solutions describe the correction curves. Each such equation contains a single numerical parameter representing "how much" misperception is taking place at each angle in the illusion, which we call the *strength* of the illusion. Some of the factors affecting the strength parameter, within the drawing, within the visual system, and related to the manner in which the drawing is viewed, are known, but its precise relationship to neurophysiology has not yet been determined. Values of the strength parameter for each of the corrected illusions shown here were determined empirically, by drawing the corrections for several closely-spaced values and picking the one that looked best. (A much more intricate mathematical model that supports the Brentano Hypothesis and is based directly on the physiology of the eye has been given by E.H. Walker(4).)

Our technical paper (3) was illustrated with classical illusions, much studied in the psychological literature, and associated with the names of Poggendorff, Zollner, Hering, Orbison, Ponzo, and Muller-Lyer. The illusions presented here are unconventional, but each has a purpose in supporting the Brentano model. The model is based on work of W.C. Hoffman (1), but with some specific differences of detail that contradict Hoffman's theory of perception. In particular, Hoffman's theory led him to predict the possibility of illusions with hyperbolic or spiral distortion patterns (see Figure 1), but not with sinusoidal distortion patterns or distorted curves (see Figures 2, 3, and 4). Indeed, Hoffman has since stated (2): "(A) spiral, representing a combination of size and rotation constancies, may be involved in such a visual illusion, but a sinusoid, for example, cannot."

Figure 1 is a variant of the classical Orbison illusion, in which a square is drawn in a pattern of concentric circles. In that case, the sides of the square appear to be bowed inward, whereas they appear to be bowed outward when the distortion pattern consists of rectangular hyperbolas. In Figure 1 (b), the sides of the "square" appear to be straight, but they are actually bowed inward, as you can verify by holding the page at eye level and sighting along one of the lines.

Figure 2 is a variant of the familiar Poggendorff illusion, in which a diagonal line is interrupted by two parallel lines. The diagonal appears to be parts of two different lines. Here the interrupted curve is a sine wave, specifically, $y = \sin 2x$. Because of the angular intersections with the parallel lines, the two portions of the curve do not appear to be smoothly connected "behind" the vertical strip (unless you look along the curves at eye level). The possibility of this illusion was specifically denied by Hoffman in (2). Figure 2 (b) was drawn with $y = \sin 2x$ up to the peak on the left, then with $y = \sin (2.1745\,x - 0.13705)$ from the peak to the left-hand vertical, with the right hand curve being a mirror image. (Because of the nonlinear equations involved, the computer was also needed to find the right numbers.)

Figures 3 and 4 use Orbison-type distortion patterns composed of closely-spaced sine waves. In Figure 3 the curves have equal period and varying amplitude, while in Figure 4 the curves differ by vertical displacement. The

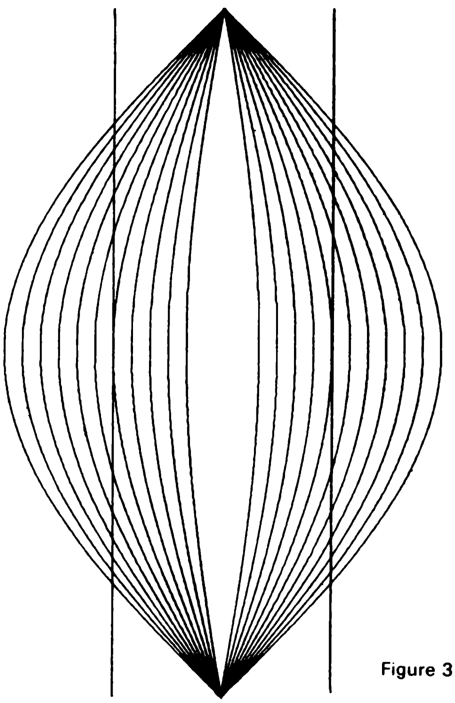


Figure 3 (a)

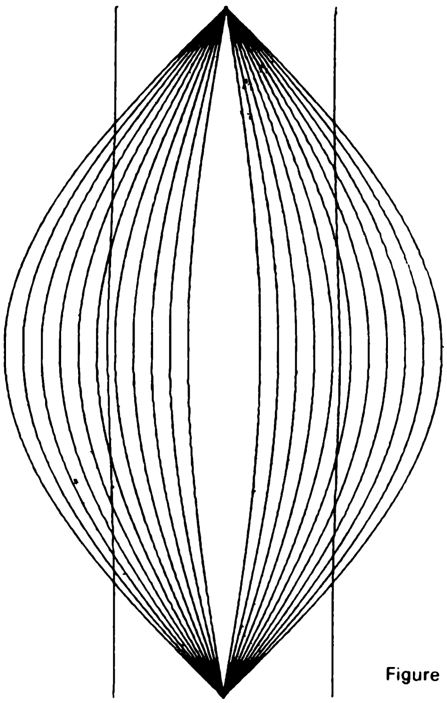


Figure 3 (b)

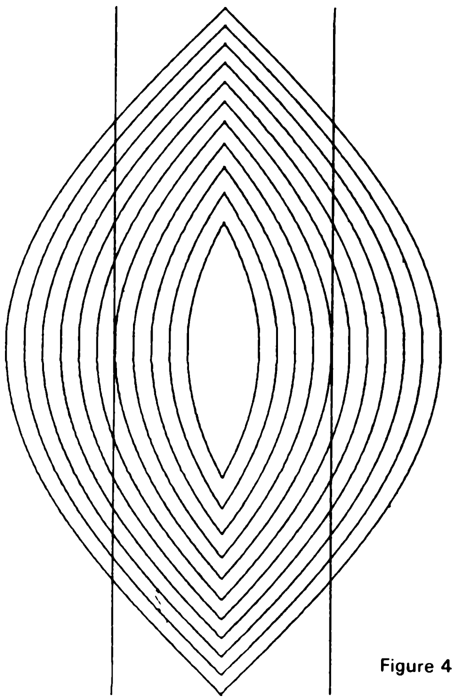


Figure 4 (a)

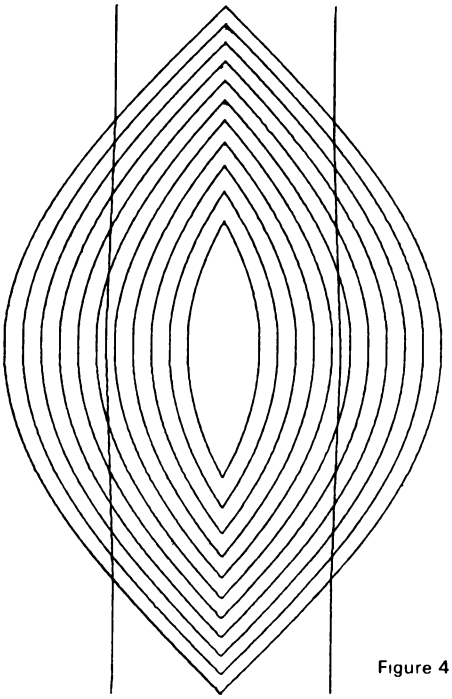


Figure 4 (b)

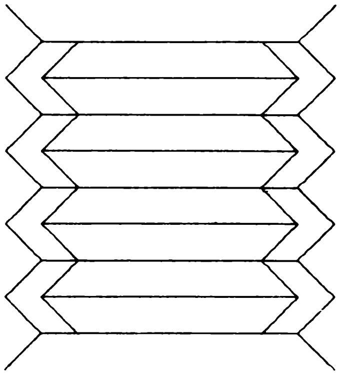


Figure 5 (a)

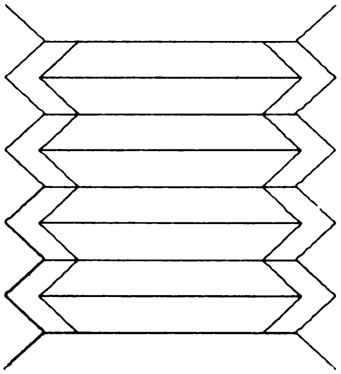


Figure 5 (b)

correction curves are produced in the same manner as those for Figure 1, by numerical solution of a differential equation.

Figure 5, which we call the "oriental lantern," is a variant of the Muller-Lyer "arrowhead" illusion. It illustrates how misperception of angles can create an illusion of *length,* as opposed to shape or angle. In Figure 5 (a), all of the horizontal lines have exactly the same length, but the five "outer" lines appear longer than the four "inner" ones. In Figure 5 (b), while the horizontals *appear* to be of the same length, the "inner" ones are somewhat longer, as you can verify by laying a straightedge on the figure. In this case, the correction is done by adjusting the slopes of the diagonal segments according to the Brentano Hypothesis, and then filling in the horizontals. ■

The drawings in Figure 1 were done on a Complot plotter at Duke University Computation Center, and the remaining figures were done on a Calcomp plotter at Chi Corporation, a subsidiary of Case Western Reserve University.

REFERENCES

1. W.C. Hoffman, "Visual illusions of angle as an application of Lie transformation groups," *SIAM Review* 13 (1971), 169-184.
2. W.C. Hoffman, "In defense of LTG/NP," to appear (presented to the American Mathematical Society, August 19, 1975).
3. D.A. Smith, "A descriptive model for perception of optical illusions," submitted to SIAM Review.
4. E.H. Walker, "A mathematical theory of optical illusions and figural after-effects," *Perception and Psychophysics* 13 (1973), 467-486.

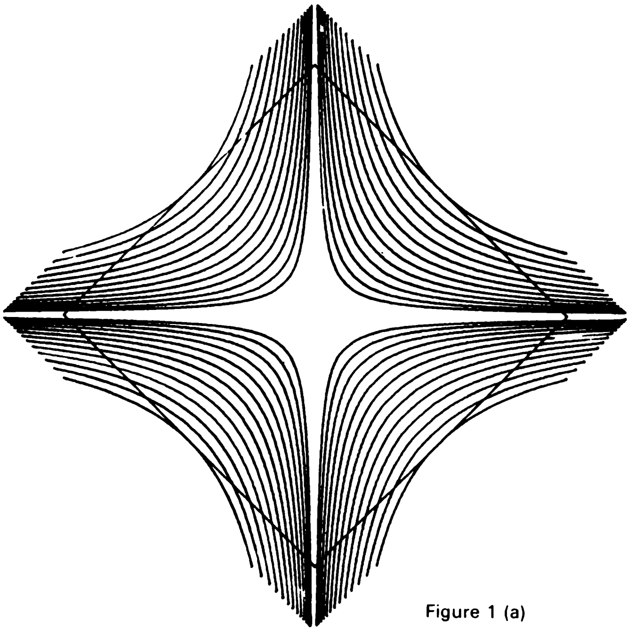


Figure 1 (a)

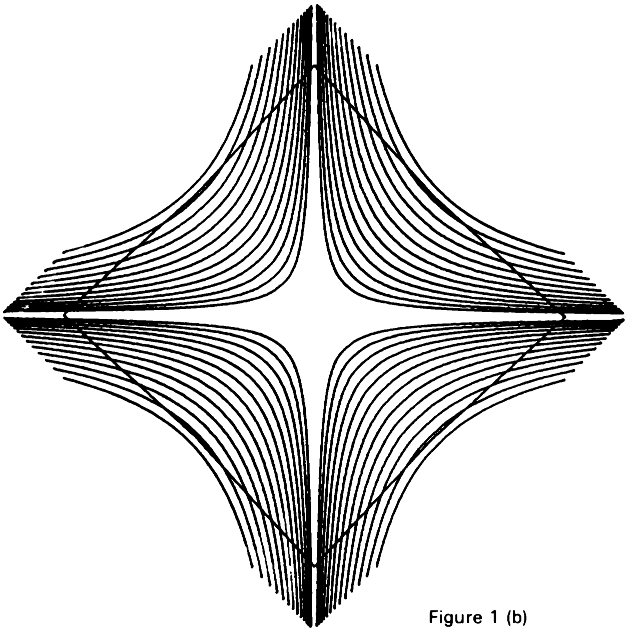


Figure 1 (b)

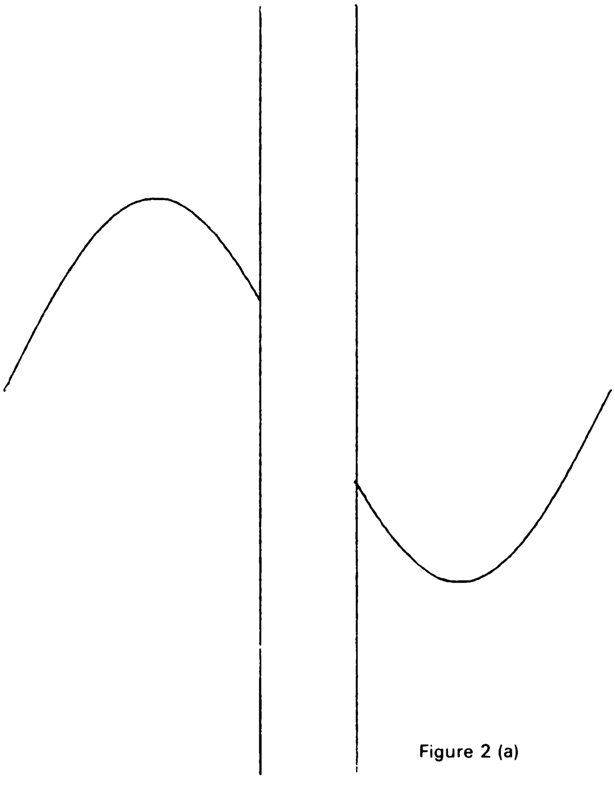


Figure 2 (a)

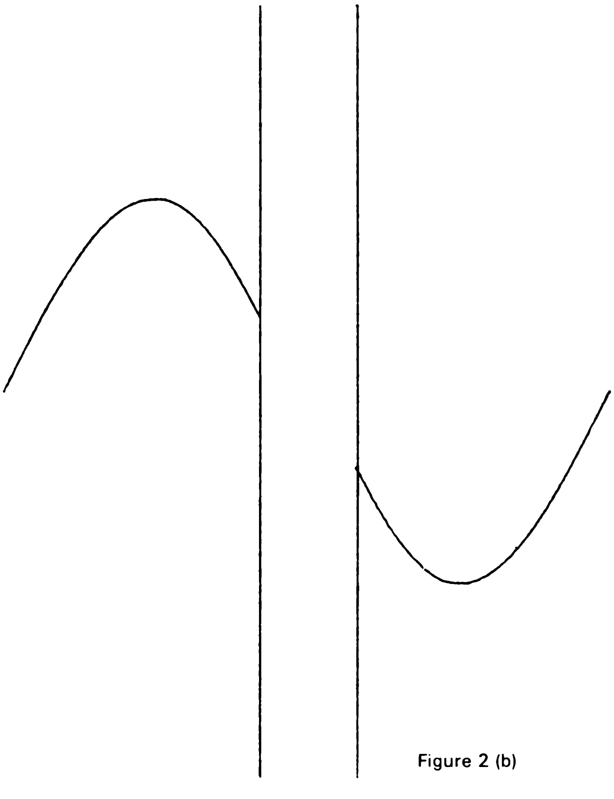


Figure 2 (b)

BROWN SCIENTISTS PEER INTO FOURTH DIMENSION

Curt Norris*

Two Brown mathematicians have succeeded in penetrating the realm of the fourth dimension through the use of very sophisticated computer graphics techniques. The unique studies, which allow the scientists to visualize the unseeable, are conducted by Thomas F. Banchoff, associate professor of mathematics, and Charles M. Strauss, assistant professor of applied mathematics.

The two Brown University professors are among the very few people in the world who are able to manipulate pictorial representations of four-dimensional concepts. They create their art by instructing the computer to investigate geometric objects that cannot exist in our three-dimensional world. The computer then produces images on a television screen in the Brown University Computing Laboratory.

As the professors twist a "stick" resembling an aircraft control, the cage-like image rotates and tumbles like an architectural model in the grip of robot arms. Then a turn of a dial sends the geometric form through rotations in four-dimensional space, producing sweeping changes in the shape of the figure.

"Students in my mathematics courses and my freshman seminar on the Fourth Dimension say that they react entirely differently to higher dimensions after becoming familiar with these representations," Prof. Banchoff reports. "The possibilities of these techniques for mathematical research are very exciting to the scientists who have seen these demonstrations," says Prof. Strauss.

Prof. Banchoff said that a good analogy of a fourth dimensional concept would be the lifetime of Abraham Lincoln. No one three-dimensional reconstruction of a historical moment would give the complete picture of the man, but a collection of all such three-dimensional pictures moving in time would give a four-dimensional portrayal of the events of the lifetime.

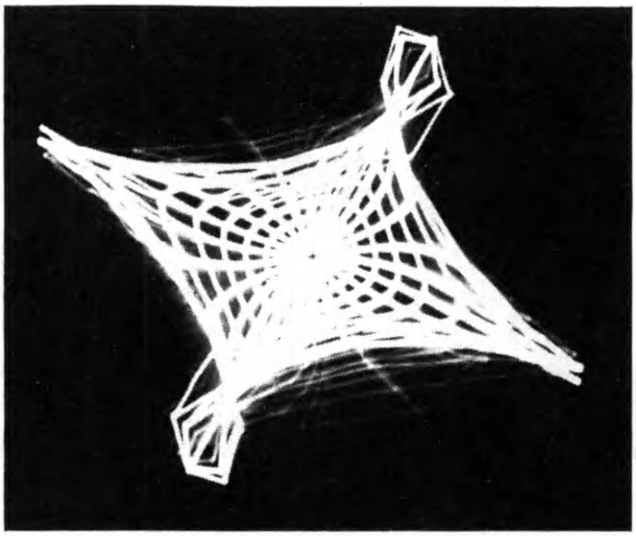
"In a similar way," the mathematician explained, "A segment—a line—moving in a direction perpendicular to itself would describe a square region, and a square moving in a direction perpendicular to itself would generate a three-dimensional cube. We could think of the cube as the lifetime of a square, with any individual square slice representing first one event in that lifetime." The pictures on the television screen show how the slices fill out a perspective view of a cube.

The next step moves into the fourth dimension.

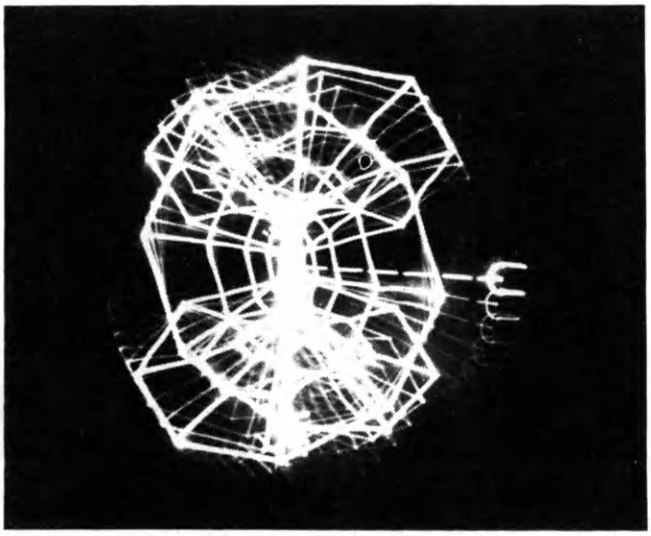
"In three-dimensional space, we can't actually move the cube in a direction perpendicular to itself to generate a four-dimensional cube, or 'hypercube,' but we can see what the images of such a fourth dimensional would be as it passes through our three-dimensional space. At the turn of three control dials, the picture of a cube begins to move away from itself, tracing out a network that unfolds and collapses back in on itself. "We are actually watching the shadows of a four dimensional cube projecting down to our space."

The hypercube is one of the least complicated of the objects that Prof. Banchoff and Prof. Strauss are studying. Some are difficult enough to require the use of a stereoscope so that the views of these projections and slices appear truly three-dimensional. Many of these ideas have potential usefulness in physics or economics, wherever quantitives of data occur which cannot be handled easily by ordinary three-dimensional representations. In many ways, the project is just beginning.

"One of our recent techniques involves describing an object by its slices," explains Prof. Banchoff. "A knife moving across a square produces a collection of parallel segments. A cutting plane moving across a cube and parallel to one face will produce a collection of squares. You might say that the cube is described as a stack of square slices, as a square is a stack of segments. In our work, we study the individual three-dimensional slices that will stack up in some direction perpendicular to all of our space to form a four-dimensional object." ■



Algebraic curve in fourth dimension



Projective plane (surface in four-space)

---

*166 E. Main St., Norton, MA 02766

An Inexpensive Reading Machine For The Blind

John M. Brus

This September a student at the Watertown, Mass. Perkins School for the Blind will put a textbook face down on a glass plate and a machine will 'read' the words to him.

The machine is a minicomputer based system that converts printed text—of any typeface—into spoken language at rates up to 200 words per minute. And it is the product of years of effort by Raymond Kurzweil, 28, and his small Cambridge, Mass. computer firm.

Blindness, affecting about 1.7 million Americans, frequently stems from disorders which can leave an individual multiply disabled. Consequently, a large percentage of the blind can never learn the Braille language. Add the fact that at most only three per cent of the books published in the United States are ever transformed into recorded form and the need for Kurzweil's machine becomes clear.

The U.S. Government agrees and is now financially backing the completion of the machine's development through the Veteran's Administration and the Bureau of Education for the Handicapped. The National Federation for the Blind is also helping financially and the machine will undergo field testing for the next 18 months before being placed on the market.

Kurzweil has combined engineering and programming ingenuity with the shrinking prices of semiconductor components to produce a product which he predicts will only cost $5,000 in five years.

How Does It Work?

Currently using a Data General Nova 2 minicomputer, the machine employs a series of programs using Assembler language to identify the individual letters and transform them into recognizable speech. "What we've done that's new is produce multiple typeface character recognition in an inexpensive minicomputer system," Kurzweil explains.

The basic sequence of operations begins when the camera—using an integrated scanning array—scans the letters in each word and feeds the data directly into the processor. A 'shape analysis' program identifies the letter and converts it into an eight-bit ASCII code. If necessary, other programs using contexual and other clues assist in the identification. Sequential algorithms then take the ASCII coded information and, using phonetic rules, transform the letters into phoneme signals. A phoneme is the sound associated with each letter. These signals in turn drive the Votrax speech synthesizer circuits.

Hundreds of typefaces are available to printers today and all vary according to line thickness, size and serifs. To deal with this variety, Kurzweil devised what he calls a 'Topological Analysis Program.' It recognizes that each letter has certain invariant relationships of shape and the program matches each letter to certain assigned 'property sets.'

For instance, Kurzweil defined an upper case "A" as a loop, a North-Central to South-West line segment, a North-Central to South-East line segment and one South concavity. Thus, the property sets are defined totally by lines, loops, concavities, vertices and their relationships. And on the average, five binary property sets define each letter.

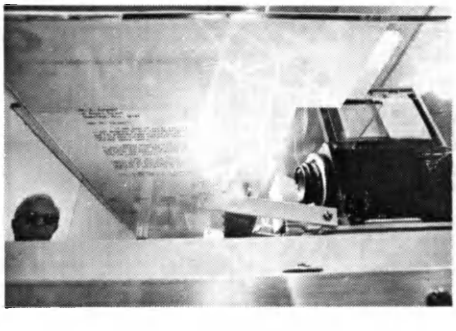
Sometimes this initial program cannot make a final identification. Then the output is an ambiguity code indicating several possible identifications. For example, a vertical bar might be an "l" or an "i" or the number "1." A back-up program makes final judgment using clues such as letter size, positional factors, presence of dots above or below and contextual clues.

Kurzweil gives an example of a contexual clue this way. "A vertical bar with a space to left and a consonant to the right is probably a capital "I" rather than a small "l" because words starting with "l" do not have second letter consonants."

If sloppy printing joins two letters, another program can identify the likely split points for the touching letters by analyzing the vertical-horizontal ratios of the image.

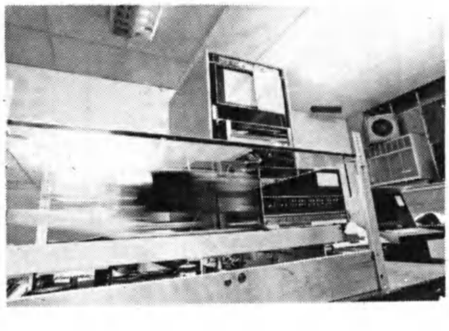
Word Pronunciation

After identifying and storing each letter, determining word pronunciation was the next programming problem. And since most of the processor was occupied by the character recognition work, there wasn't enough memory remaining for a standard 100,000 word English dictionary.



The basic mode of operation of the Kurzweil Reading Machine is automatic. The user simply places printed reading material face down on the glass plate of the desk top reading unit and presses a button to begin the scanning.

James Gashel, chief of the Washington office of the National Federation for the Blind, operates the simple controls of the Kurzweil Reading Machine, the world's first multi-font, full word speech reading machine for the blind.



In this time-lapse shot, the "scanner" converts a typewritten letter into digital signals for analysis by the computer in the electronic control unit (right), which contains the character recognition subsystem, the speech subsystem, and the synthesizer.

Raymond Kurzweil, president of Kurzweil Computer Products and inventor of the Kurzweil Reading Machine, points to the machine's "scanner," a small electronic camera moving on linear bearings and transmitting light and dark images of a printed page to the computer in the control unit.

Kurzweil s solution was programming a set of 1,000 phonetic rules for the English language and an additional 2,000 entry dictionary for exceptions. Both rules and exceptions were "practical for a minicomputer based system since the rules only required about 4,000 16-bit words of storage," according to Kurzweil. And to increase the effectiveness of the rules and exception dictionary, he added another program that can strip off prefixes and suffixes.

A final speech system provides a 'stress contour' across each sentence. He says this reduces the monotony of the speech and is intended to reduce listener fatigue. The synthesizer circuits give the machine's voice a thick Swedish accent that takes getting used to, but Kurzweil says most people can readily understand the voice with a couple hours practice.

However, the system cannot differentiate the pronunciation of about 20 'ambiguous' words. For example, 'lead' in 'lead magnet' pronounces differently from 'lead' in 'You lead, I'll follow.' But Kurzweil cites cost as the rationale for dismissing the problem, not technical difficulty.

Once all the bugs inherent in such a complex system are resolved, he plans to market the machine initially to institutions and then to leasing companies catering to individuals. ■

Medical Computerized Data Bases

by Susan Hastings

The computer stands at the threshold of the entire health care industry. It already controls many administrative functions. Doctors are using it to make diagnoses and to simulate surgical procedures. Comprehensive medical data banks have already been established in this country. With national health insurance just around the corner, computer use is exploding throughout the healthcare field.

Those who object to the increasing use of computer systems in medicine say that it will destroy some of our most sacred institutions, including the confidentiality of the traditional doctor-patient relationship. They believe that medical computers pose a great threat to privacy because they contain information required by so many other agencies; they feel that once patient records are put into computers which will make information accessible from remote terminals, there may be no limit to violations of privacy.

We should remember, however, that the computer itself is only a tool. And while tools can hurt us if not used sensibly, by definition, their prime use is to benefit man in controlled situations. Technology is continuing its search to develop devices that will improve the control man holds over his newest, and maybe most powerful tool, the computer.

"Security" is the technological term given to the devices that will help to insure the privacy of information once it is contained in a data bank. The computer is capable of providing security to the medical community in safeguarding its records and its traditions of confidentiality, and the new government legislation will regulate the kind of information that may be collected in data banks. Efforts like these to insure a patient's privacy actually make the medical information stored in a computer seem less vulnerable to invasion than information stored in a doctor's file cabinet or on a medical chart in a hospital.

The Placebo and the Computer— Unexpected Antagonists

A few years ago, a drug called reserpine became available for treating homicidal patients. St. Elizabeth's Hospital in Washington set up a trial to test the effectiveness of the drug. One group of homicidal patients was to receive the drug. Another group of homicidal patients was to receive a placebo sugar pill, which purportedly has no pharmacological effectiveness. Neither patients nor doctors involved in the study knew which of the two groups of patients would be getting which pill.

The psychiatric resident who gave medication to one of the two groups was Werner Mendel (now a professor of psychiatry at the University of Southern California School of Medicine). Shortly after the drug trial started, Mendel became convinced that his patients were receiving reserpine because they calmed dramatically. The more convinced he became, the more they improved. After the study was over, however, he learned that his patients had received the placebo. And that's when the power of the placebo first hit him. If a physician believes in a medication, he decided, he will transfer that belief to his patient, and his patient's condition will improve.

Many physicians, however, do not appreciate the power of the placebo and what it can do for patients. This charge is leveled in the June 23 *Journal of the American Medical Association* by Herbert Benson and Mark D. Epstein, physicians at Harvard Medical School.

There is ample evidence, report Benson and Epstein, that placebos can help patients with a variety of ills — pain, heart attacks, rheumatoid arthritis, hay fever, headache, cough, peptic ulcer, anxiety, depression. It is precisely because of this evidence, they argue, that physicians should look into how placebos work and exploit them to full advantage.

Only when physicians better understand the scientific basis for placebo effectiveness, Benson and Epstein conclude, will they be able to incorporate advantageously the placebo into evolving forms of health care. They are concerned that taking patient histories by computer will not allow physicians to develop the rapport with patients that is necessary to heal them. ■

Music Dream Machines:

New Realities for Computer-Based Musical Instruction

by
Fred T. Hofstetter*

Even though it is still a technological infant, the computer has emerged as a powerful medium with the potential of solving some of the most complex problems faced by today's music educators. Courses of study in computer technology are now a common part of the graduate music curriculum. The growing membership of the National Consortium for Computer-Based Musical Instruction indicates that a concerted effort is under way to realize the potential of computers in the music classroom. Moreover, the fact that the Norlin Music Company, which makes Moog synthesizers, Lowery organs, and Gibson guitars, is manufacturing a stand-alone music education system using minicomputer technology indicates that a major trend in music education may be under way.

At least twenty university music departments have initiated major developmental programs to produce computer-based musical materials (Jones, 1975). Preliminary reactions to these materials have been overwhelming. Teachers like them; students like them; and researchers like them.

The classroom music teacher faces many logistical problems. A well-planned presentation usually involves the use of several media, such as records, tapes, slides, transparencies, and duplicated handouts. The physical preparation of these materials, as well as the actual manipulation of them during class, can be cumbersome and time-consuming for both teacher and student. The teacher can encounter difficulties in procuring desired materials for the presentation, and unless time is taken to locate the material, the students have to settle for a second-rate presentation. Students remember many occasions when valuable class time was lost while the teacher tried to find the right place on a record or a tape in order to play a musical example.

If a music teacher were asked to state the requirements of a classroom presentation "dream-machine," the response would be a device capable of displaying musical notation, showing slides, playing recordings, and maybe even generating some new examples for aural training. Such a machine was dreamed about in the 1960's by Professor Donald Bitzer (1961) at the University of Illinois. He made it a reality; it is called PLATO, and it is now a product of the Contral Data Corporation (1976). PLATO is a comprehensive computer-based education system which has been used in over seventy-one subject areas, and a variety of special presentation devices are available in different subjects.

For all subjects there is a basic PLATO display unit which contains a screen upon which graphics (like musical notation) can be drawn, a random-access microfiche projector which can show slides on the screen, a typewriter keyboard through which one can communicate with the computer, and a touch panel which allows students to answer questions by touching pictures or words on the screen. Two special devices are used in music programs. One is a random-access audio device which can play any segment of a pre-recorded magnetic audio disk, and the other is a four-voice synthesizer which can be played by the computer. All of these components combine to produce a single machine which can display musical notation, show slides, play recordings, and generate new aural examples.

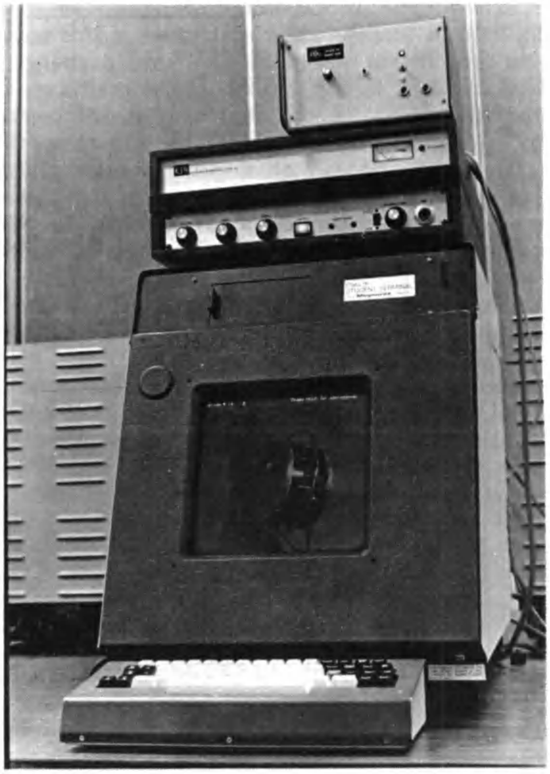


Figure 1. PLATO terminal with microfiche slide projected on the display screen. On top of the terminal is a random access audio unit, and the small box on top of that is the 4-voice synthesizer.

---

*The University of Delaware, Newark, DE

One of the greatest benefits of computer-based education is the return of personal warmth to the classroom made possible by releasing the teacher of mechanistic duties.

A super-medium like PLATO can put an end to the problems of finding the right materials because vast numbers of recorded examples, slides, and visual displays can be stored and randomly accessed at a split-second's notice in the computer. Because it has the ability to present visual displays, play aural examples, ask students questions, record student responses, and interact with students on the basis of their responses, the computer is an ideal medium for drill and practice. Due to the limited amount of time the teacher spends with students in the classroom, it is difficult to maintain a good balance between concept development and drill-and-practice activities. In trying to meet course objectives by their intended deadlines, teachers realize that their students often do not get enough practice. Teachers also know that when they devote time to classroom drills, the individual differences among students create an environment in which some students are bored because they have already mastered the material, and others are hopelessly lost because the exercises are too difficult for them. It is now possible to take drill-and-practice out of the classroom and have it done in the computer laboratory where each student receives an individualized course of study based on the student's personal learning needs. And by means of the data keeping facilities of the computer, the teacher can get periodic reports on the progress and special problems of each student.

When drill-and-practice is done in a self-paced computer-based environment, learning is more efficient. Class time formerly devoted to drills can be spent in concept development and in more creative aspects of music. The role of the teacher becomes more human. Mitzel (1972) has noted that one of the greatest benefits of computer-based education is the return of personal warmth to the classroom made possible by releasing the teacher of mechanistic duties.

The greatest proponents of computer-based musical instruction are the students themselves. It has been demonstrated that with the aid of the computer they can increase their musical sensitivities, make higher grades, spend less time completing course requirements, and rediscover the intrinsic joy of learning. Illustrations of these attributes can be seen in several schools throughout the country. At Penn State University, Diehl (1971, 1973) demonstrated that students can develop more sensitivity and accuracy in the recognition and performance of articulation, phrasing, and rhythm. An experiment was set up in which 25 junior high school students who had taken private woodwind or brass study for at least three years were tested on their sensitivity and ability to perform articulation, phrasing, and rhythm. After this pre-test they participated in a five-week computer program in which an IBM 1500 computer was used to display musical notation, play pre-recorded musical examples, and ask questions about the articulation, phrasing, and rhythm of the musical examples. The students used the computer one hour each week. At the end of the five-week program, the students were given the test again. Table 1 gives the mean test scores for the pre-and post-tests. It is interesting to note that the computer led to a 55% gain in the students' abilities to *perform* music.

TABLE 1

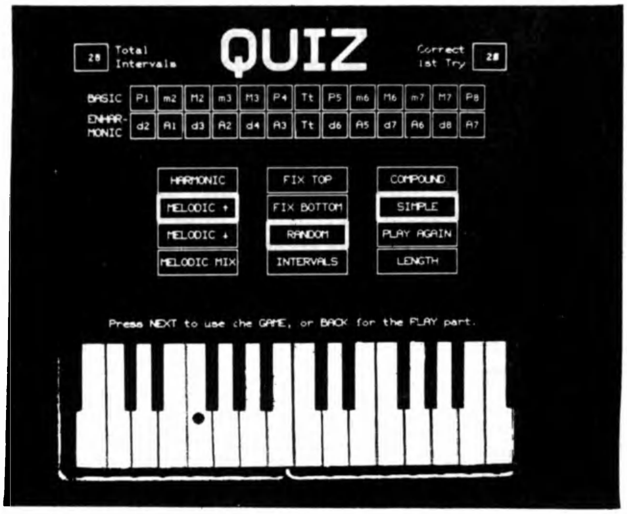
Summary of Student Scores in a Computer-Based Program for Improving Articulation, Phrasing, and Rhythm of Intermediate Instrumentalists

Group Averages	Listening Test Means	Performance Test Means
Pre-Test (N=25)	62.8%	38.8%
Post-Test (N=25)	92%	94%
Gains	30%	55.2%

At the University of Delaware, a computer-based ear-training system has led to significant improvement of instruction in core music theory courses. Ear-training students are scoring a full letter-grade higher than they did before. The system is called GUIDO (Hofstetter, 1975), named after the eleventh century monk and music educator who invented the staff and the solfeggio syllables *do, re, mi, fa, sol,* and *la.* GUIDO is a acronym for *G*raded *U*nits for *I*nteractive *D*ictation *O*perations. These units are stored in the computer, and they contain a complete curriculum for aural drill-and-practice in intervals, melody, harmony, rhythm, and chord-qualities. GUIDO is being implemented on a variety of computers. To date, its best operating environment is on the PLATO system, because PLATO's touch-sensitive display screen makes it possible to run the program by merely touching musical symbols on the screen. It is not necessary to type on the keyset.

Figure 3 shows a sample display from the intervals program. By studying this display the basic features of the GUIDO system can be understood. At the top are two rows of boxes which contain the names of musical intervals. When the student wants to hear an interval, all he has to do is touch one of the boxes. When he does, the box lights up and the interval designated by the box is played by the computer-controlled synthesizer. Conversely, when the student is going through one of GUIDO's formal units, the computer plays an interval, and the student responds by touching the box which contains the interval he thinks was played.

Figure 2. Professor Ned Diehl observes a clarinet student using the Penn State Instrumental Music System.



Press NEXT to use the GAME, or BACK for the PLAY part.

Figure 3. Touch-Sensitive Display from the GUIDO Ear-Training System.

Underneath the interval names are three columns of teacher or student control boxes. These boxes are used to control the way in which dictation is given. The teacher can preset them for the students, or the teacher can allow the students to set them at will. The first column of boxes allows for the intervals to be played as harmonic, melodic down, or melodic intervals up and down. The second column gives the option of being able to fix the top or bottom note of the intervals, or to have them selected at random. The box marked "intervals" allows the student to eliminate intervals from the boxes at the top of the screen, so that only some of the intervals will be played. In the third column of boxes the student can select compound or simple intervals, can have an interval played again, and can change the length of time the intervals last. Finally, there is a keyboard at the bottom of the screen. When intervals are played in formal units one of the notes of each interval is shown on the keyboard, and the student is asked to touch the other note played in the interval. In this way, students are quizzed on the spelling as well as on the aural recognition of intervals.

In addition to liking computers because they make learning more efficient and improve grades, students like the interactive help which the computer gives them in analyzing music. One example is the combination of tutorial and analytical programs being developed on PLATO by Wittlich (1976) at Indiana University. These programs are based on *set theory*, the analytical procedures defined by Allen Forte (1973) for describing the pitch organization of atonal music. Wittlich is writing programs to teach music fundamentals using pitch class concepts of set theory. Topics include the definition of set, normal form, numerical set, transposition, simple inversion, inversion relative to zero, transposed inversion, interval class, and interval vectors, A very time-consuming part of set theory is the computation of normal form, interval vector, and set name for each pitch set. In fact, many people do not use these procedures because of the time involved. James Trueblood, a former Indiana student and now a programmer at Delaware, has written a program which allows the student to type in the pitch set using either letters or numbers, and then, in less than .3 second, the normal form, vector, and set name are shown on the display screen.

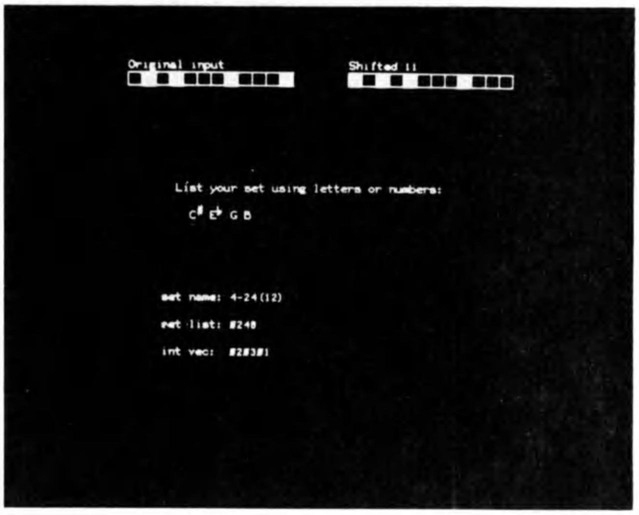
Figure 4 contains a sample display from the set-theory program. In the middle of the display can be seen where the computer asked for a pitch set. After the student typed in C# Eb G B, the computer immediately responded with the rest of the display which shows that the normal form is 0 2 4 8, the interval vector is 0 2 0 3 0 1, and the pitch-class name is 4-24 (12).

> **The computer-based music education system can present visual and aural stimuli, record responses, and perform data analysis in one automatic process.**

Students also like the control which the computer gives them in writing compositions for electronic music synthesizers. Instead of struggling with dials, switches, and patch cords, the students can write computer programs which automatically control the synthesizer and create desired sounds with much less effort. In this way students can concentrate more on the organization and meaning of their music. An example of this application is the Iowa State Computerized Music System (ISMUS), developed under the direction of Gary White. This system allows students in music, computer science, and engineering to interact in the creation of musical compositions played on an ARP synthesizer under the control of a PDP-8 minicomputer.

The last and most obvious reason why students like computer-based education is because it is just plain fun. Even the circle of fifths can be fun. Figure 6 shows the display screen of a game developed under the direction of Professor David Peters (1976), head of the PLATO muisc project at the University of Illinois. Named "Keyspinner," this game calls for two players. The display consists of the circle of fifths with a needle or "spinner" inside it. The players take turns spinning the needle, and when it stops on a key signature (e.g. 5 sharps), the player has to identify the key of that signature (in this case B) as soon as possible. The computer keeps track of how long it takes for each player to answer correctly, and the player with the best score wins.

Musicians have also found the computer to be a powerful tool for pedagogical and experimental research. In the past attempts to measure the effectiveness of instructional strategies and the nature of musical learning have been hampered by the need to use manual techniques for saving and recording data. The amount of time which this requires per subject has resulted in the use of small populations when large populations would have been desired, and it has limited the number of experiments which music educators have been able to do. Of course, computers have been around for quite some time, and musicians have been using them to analyze data. What is different is that whereas in the past experimenters had to present stimuli and manually record and encode responses for data analysis, now the



```
Original input                 Shifted 11

      List your set using letters or numbers:
        C♯ E♭ G B



     set name:  4-24(12)

     set list:  0240

     int vec:   020301
```

Figure 4. Sample display from Jim Trueblood's set theory program

Figure 5. Professor Gary White (seated) composing on the Iowa State Computerized Music System

computer-based music education system can present visual and aural stimuli, record responses, and perform data analysis in one automatic process. This means that the amount of experimentation can be greatly increased. It means that music educators can spend more time asking questions, and less time shuffling papers.

Another advantage is the exact control which the computer has over the sound source. If the researcher wants to play a sawtooth wave at 50 db for .3 second, he can accomplish this by giving the computer a few simple instructions. This immediate, precise control of the sound source reduces the time needed to set up an experiment, and it guarantees that during the course of an experiment there will be no unintentional variation of the sound source which could bias the results.

To date, the best example of the use of a computer-based music education system for experimental research is the study of harmonic and melodic interval recognition conducted at Stanford by Killam, Lorton, and Schubert (1976). Using a PDP-10 computer, a model 33 KSR teletype, and a computer-controlled Thomas model 145 solid-state organ, a system was designed to play intervals, ask subjects for the names of intervals played, and record their responses. Six sets of intervals were played for each of fifteen subjects. One set consisted of 48 harmonic intervals played for .2 second. The second set consisted of 48 ascending melodic intervals played for .2 second. The third set consisted of 48 descending melodic intervals played for .2 second. The fourth, fifth, and sixth sets were the same as sets one, two, and three except that they were played for only .1 second. Within each set, intervals were selected at random, and they ranged in size from a minor second to a perfect octave.

The results of this study showed that some beliefs on which teaching methods are based may be myths. First of all, when students encounter difficulty in hearing intervals in classroom dictation drills, one of the first techniques the teacher will try is slowing down the speed of dictation. However, in this study there was no significant difference in student performance when intervals were played for .1 second, when the average correct recognition was 76%, or for .2 second, when the correct recognition was 77%. Second, it is commonly believed that the perfect octave is extremely easy to recognize, that it is such a "sure thing" that little class time needs to be spent studying it. In the Stanford experiment, however, it was shown that the perfect octave is not such a sure thing. In fact, it was missed an average of 12% of the time. Moreover, seven subjects found the octave more difficult to recognize than some other interval. Finally, the study showed that there are differences of recognition related to the mode of presentation used. Table 2 contains a confusion matrix summary for modes of presentation. In this table it can be seen that the most frequently used wrong answers vary as a function of mode of presentation. For example, whereas the subjects confused the minor sixth (m6) with the perfect fourth (p4) in simultaneous and ascending intervals, they confused it more often with the major sixth in descending intervals.

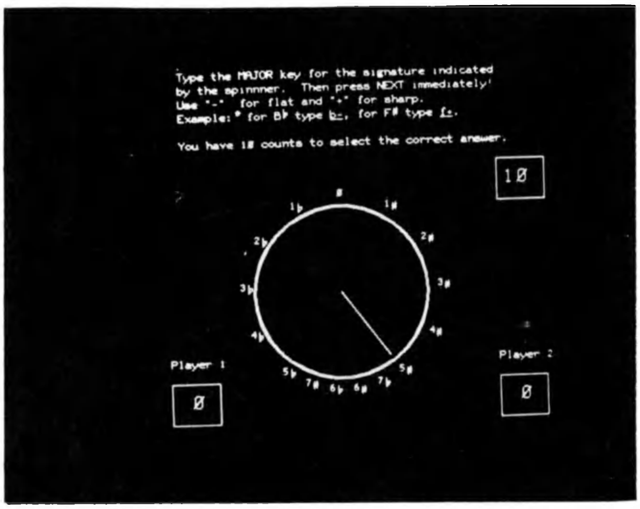
Information about student learning patterns is needed by writers of textbooks so that their materials can be presented in the best order, by teachers so that they can deal more effectively with their students, and by the students themselves so that they can be aware of and try to avoid common pitfalls of musical perception. With the emergence of the computer-based music education system a comprehensive body of knowledge about musical learning will be obtained whereby these needs can be fulfilled.

It has been shown how computer-based techniques are improving music education for the teacher, the student, and the researcher. They have generated such wide-spread interest that vendors are beginning to market programs for music instruction. It was mentioned that the Norlin Music Company is manufacturing the first stand-alone music education system. It is doubly encouraging to see this support from vendors at a time when the cost of computer hardware is rapidly decreasing. One does not need to go out

TABLE 2

Confusion Matrix Summary for Modes of Presentation
in the Stanford Study of Interval Recognition
Most Frequently Used Wrong Answers

Stimulus	Simultaneous	Ascending	Descending
m2	M2	M2	M2
M2	m2	m3	m2
m3	M3	M3	M3
M3	m3	m3	m3
P4	M3 & P5	P5	P5
T	m3 & M7	P4	P4
P5	P4	P4	P4
m6	P4	P4	M6
M6	m6	m7	m6
m7	M7	M7 & M6	M7
M7	T	m7	m7
P8	M7	M7	P5



Type the MAJOR key for the signature indicated
by the spinnner. Then press NEXT immediately!
Use "-" for flat and "+" for sharp.
Example: for B♭ type b-, for F# type f+.

You have 10 counts to select the correct answer.

Figure 6. Display of the Keyspinner game developed by Professor David Peters at the University of Illinois.

on a limb to predict that broad-based implementation of computer-based musical instruction will occur before the end of the next decade. ■

Bibliography

Bitzer, D. L., P. Braunfeld, and W. Lichtenberger. PLATO: An automatic teaching device. *IRE Transactions on Education,* E-4 (December, 1961), 157-161.

Control Data Corporation. *Control Data PLATO system overview.* St. Paul: Control Data Corporation Publications, 1976

Diehl, Ned C. Computer-assisted instruction and instrumental music: implications for teaching and research. *Journal of Research in Music Education,* 1971, *19,* 299-306.

Diehl, Ned C., and Ray H. Ziegler. Evaluation of a CAI program in articulation, phrasing, and rhythm for intermediate instrumentalists. *Council for Research in Music Education,* 1973, *31,* 1-11.

Forte, Allen. A program for the analytic reading of scores. *Journal of Music Theory,* 1966, *10*(2), 330-64.

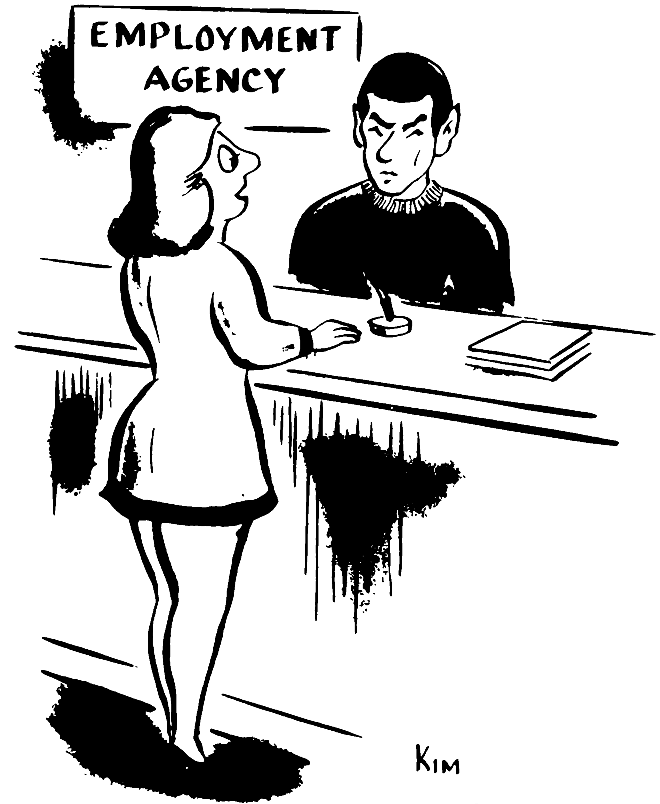
Hofstetter, Fred T. GUIDO: An interactive computer-based system for improvement of instruction and research in ear-training. *Journal of Computer Based Instruction,* 1975, *1,* 100-106.

Jones, Morgan. *Computer-assisted instruction in music: a survey with attendant recommendations.* Ph.D. dissertation, University of Iowa, 1975.

Mitzel, Harold E. "The Potential Contribution of Computers to Instruction Reform." *Alternative Futures in American Education,* Appendix 3 to Hearings on H. R. 3606 and Related Bills to Create a National Institute of Education Before the Select Subcommittee on Education, January, 1972.

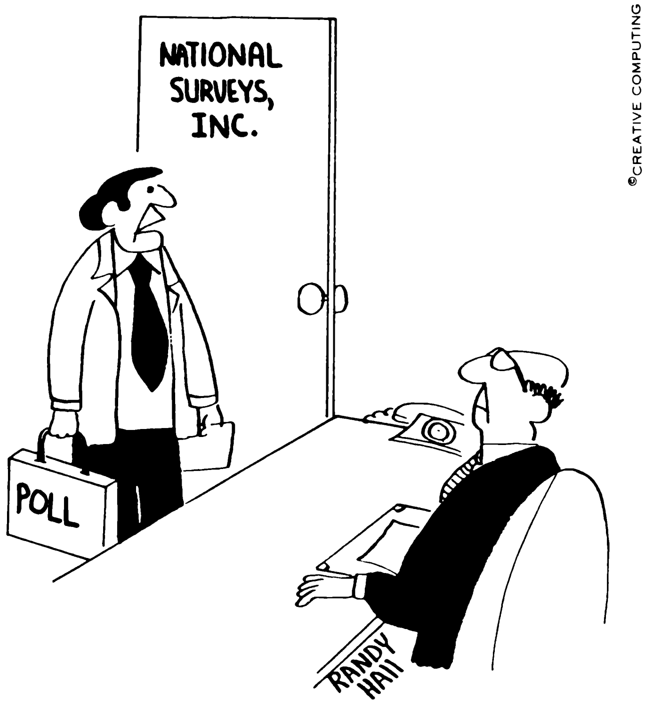
Peters, G. David. The fourth revolution in education and music: the PLATO music project. Paper presented at the Music Educators' National Conference, Atlantic City, March 12, 1976.

Wittlich, Gary. Computers and music instruction at Indiana University (Bloomington). *Proceedings of the First Annual Meeting of the NCCBMI* (University of Delaware: Music Department, 1976).



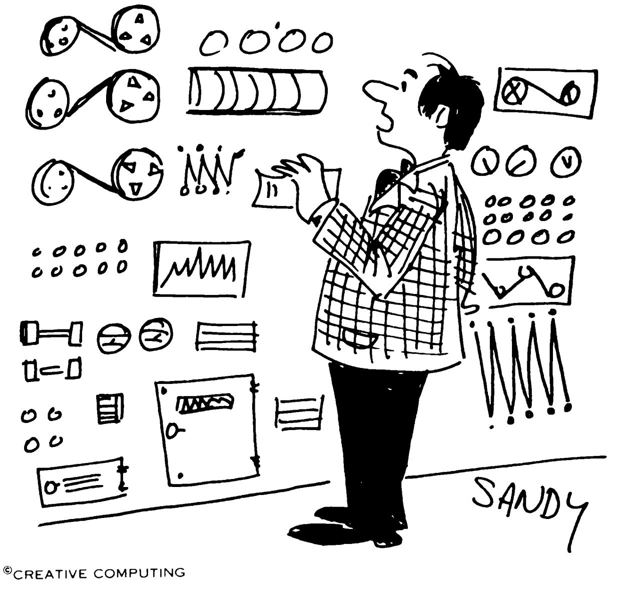
©CREATIVE COMPUTING

"You say you're an expert on computer technology?"



©CREATIVE COMPUTING

"The housewives are no longer complaining about dishpan hands. Now they've got push-button fingers."



©CREATIVE COMPUTING

"Why did you turn off my power? I paid my bill. See? Here's my canceled check."

INTERACTIVE WOMAN-MACHINE IMPROVISATIONS OR LIVE COMPUTER-MUSIC, PERFORMED BY DANCE

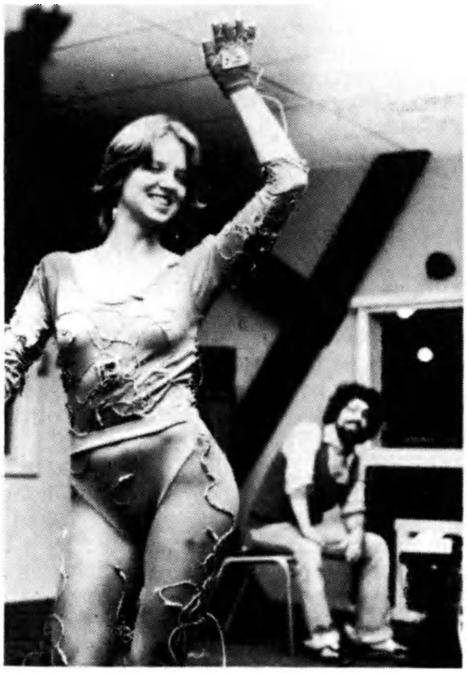
Debra Loewen is wearing a gravity-sensor costume that is monitored by a computer and correlated instantaneously with sound from a synthesizer. Depending on the user's program, Debra could be performing a new musical instrument, executing a dance piece that determines its own unique music score in real time, or experimenting with bio-feedback through physical movement and sound.

The costume has 64 mercury switches (as digital bits) multiplexed to single data lines for shipment (by cable or radio) to an interface. The system hardware — designed and built by Thomas Noggle and Joseph Pinzarrone in Urbana, Illinois, 1972-1975 — consists of a READ/WRITE bus structure between a P.D.P. 11/10 and a programmable bank of analog sound-snythesizer modules (i.e. oscillators, filters, amplifiers etc.).

System software, coded by Elven T. Riley and T. Rust, allows user control of analog device interconnection and data as well as costume information monitoring and masking and — can be used to create real-time interaction between the two.

The system has been featured chiefly in live performance from coast to coast, with Debra as interfacee in a myriad of theatrical situations including composer Pinzarrone's theatre work *Hunger Artist.* Currently, a lecture-demonstration is available entitled "Interactive Woman-Machine Improvisations." Expansion of the system, of course, remains proportional to financial support.

Debra is a choreographer-dancer on the faculty of the University of Delaware. Thomas is an engineer with High Energy Physics at the University of Illinois. Joe P. (in photo background) is a composer and co-director of the Mid-Atlantic Modern Music Institute in Wilmington, Delaware.



NEW HORIZONS FOR MICROCOMPUTER MUSIC

Malcolm Wright[1]

Since October of 1974 when the first 8-bit micro processor kit was introduced to the hobbyist, the computer kit market has exploded with a variety of supporting peripheral circuits. Who would have guessed that today a person could have his own personal computer at home to generate a form of animation on the television screen, play games in a software language like BASIC, control home appliances like a burglar alarm, or produce different frequencies to an audio amplifier in the form of music? All this, and yet few of the applications or potentials of the micro computer kit have been developed.

One of the applications for the micro computer which is just starting to be explored by the hobbyist is music. With only 45 bytes of instruction a simple routine was written by Paul Mork which could read a table of binary numbers in memory and generate a square wave frequency related to the value of the numbers. The small program by Paul could play simple melodies like "Daisy" or "Jingle Bells" when executed. Due to the speed of the micro computer, frequencies up to 2000 cycles per second could be produced.

By December 1975 the scope of software music was expanded again. Alpha-numeric music with amplitude control was introduced by *PCC*[2] in a magazine article. The author, Malcolm Wright had written Alpha-numeric music for the 8080 with capabilities not considered before. The music was still a coded table of bytes for each melody, but the bytes were alpha-numeric characters in the ASCII format. Now to play a note like middle C, the user just typed "4C" which specified the octave and the note. If the user wanted a sixteenth note of B flat, one octave higher, he would type "5SB!" into memory. Alpha-numeric music allowed the user to vary the volume, tempo, duration of the notes, generation of rests, repeat measures in music, generate six octaves of notes — sharp or flat, and create envelopes (attack) for special tonal qualities.

Software has its limits and many companies are in the prototype stage of developing computer control hardware devices for music. The modern electronic organ with a band box (rhythm generator) gives an idea of how far we can go in synthesis of instrument sound. Imagine an instrument like the MOOG synthesizer, used in many electronic music recordings, controlled by a computer! Dr. Prentis Knowlton in Pasadena, California has interfaced a PDP8 minicomputer with a pipe organ and with the assistance of many interested friends has encoded musical pieces like Bach's *Concerto in A-Minor* and Rimsky-Korsakov's *Flight of the Bumble Bee* for computer control. The Bumble Bee can be played at tremendous speeds by the computer with no mistakes and with complete repeatability.[3]

If one is going to generate electronic sounds from a special circuit board for a computer, what should be some of its capabilities? The circuit should be able to simulate different tonal qualities by generating different complex waveforms other than just sine or square waves. The circuit board should be able to give different attack and decay times for the notes to realistically simulate the various kinds of musical instruments. The frequency range of the circuit should be the whole audio spectrum from 15 cps to 20,000 cps at the minimum. The user should have control of the volume and the duration of the notes generated.

Another requirement that should be placed on the music synthesizer circuit board is that a minimum of the computer's time should be used to control the card — less than 50%. If the control time is less than 100% computer usage then the computer can be executing other programs at the same time. Imagine computer games with sound effects in the background at the same time!

The future for computer controlled instruments or synthesizer sounds is exciting. There are at least three companies presently developing these kinds of products. Solid State Music in Santa Clara, California is presently prototyping an Altair compatible card which will meet all of the above requirements.

As a last note, imagine the future composer being able to write and edit pieces of music for a whole orchestra and being able to play the music instantly after completion by typing RUN on his computer! ■

---

[1] Solid State Music, 2102A Walsh Ave., Santa Clara, CA 95050

[2] Peoples Computer Company, Box 310, Menlo Park, CA 94025

[3] An LP record of this system, "Unplayed by Human Hands" is available for $6.98 from Computer Humanities, 2310 El Moreno St., LaCrescenta, CA 91214.

BOTTOM-UP BIZET

REFLECTIONS ON IMPLEMENTING RELEASE 234.5 OF THE PEARL FISHERS

Robert P. Taylor

INTRODUCTION

Computers are wonderful devices and with them we accomplish wonderful, even astonishing things. But what astonishes me most is the freshness which computing provides into what are essentially non-computing activities. By identifying parallels with computing in a non-computing activity, I can often deepen my appreciation and understanding of a familiar human enterprise, enriching my life considerably in the process. I sometimes feel this may be a more significant reason for getting involved with computing than is the whole business of getting the computer to perform as a marvelously powerful and flexible tool in any of a host of scientific and commercial enterprises. I do not feel most computing professionals take seriously enough the importance of this "fringe benefit" of computing. In fact, I believe if we systematically encouraged and publicized the application of such insights to significant cultural enterprises, we would both enrich our culture and take a significant step toward countering the growing popular misconception of computing as a mechanistic, dehumanizing force in our society. The objective of this paper is to illustrate how this can be done in terms of one well-established and sanctified art form in our culture, grand opera. It should adequately suggest the merits of the idea.

Following this introduction, the remaining text of this paper is organized into five parts. *PART ONE* reviews the origin of the paper and likens the production of opera generally, and George Bizet's *The Pearl Fishers* in particular, to the implementation of a software system. *PART TWO* likens the New York Lyric Opera Company (NYLOC), as it was organized to produce the *Pearl Fishers,* to a software implementation team. *PART THREE* examines the "documentation" normally available for implementing the *Pearl Fishers* system and finds that it relates almost exclusively to the audio systems of the overall opera system. *PART FOUR,* using the example of the lumination system, discusses why and how all the video systems in the *Pearl Fishers* implementation project developed their own temporary, makeshift documentation. *PART FIVE* draws several conclusions, some about opera as system, some about the fruitfulness of extending the approach taken in this paper.

PART ONE: DRAWING THE ANALOGY BETWEEN OPERA PRODUCTION AND SYSTEM IMPLEMENTATION

I first became interested in the parallels between opera and systems while singing in a recent New York Lyric Opera Company (NYLOC) production of *Don Giovanni.* As we rehearsed and subsequently performed that opera, I was increasingly impressed with the parallels between producing *Don Giovanni* and implementing a payroll or other reasonably complex software system. Many of the structural relationships between the personnel producing the opera closely resembled those characteristic of a good software project team. Activities were modularized and were developed, tested, modified and integrated just as the sub-components of a software system frequently are. Success in producing the opera seemed to depend heavily upon fitting the different strands together in the right place, at the right moment, much as the successful implementation of a software system depends heavily upon interface definition and creation.

Long before opening night, I had begun to entertain myself in slack moments by trying to look at the opera production as though it were a software system implementation. The audience became end users. Cues became interfaces. Lighting and sets became sub-systems. Section rehearsals became module testing. Rehearsals became debugging sessions. The dress rehearsal became a pilot run. The voltage limitation in the lighting power source became a hardware constraint. And on it went, until by the final curtain, I had come to see that whole NYLOC production as merely the most recent implementation of the *Don Giovanni* system.

It happened that I was teaching a course on systems analysis shortly after this experience with *Don Giovanni* and I began to think that attending an opera rehearsal might be a beneficial experience for the students in that class. After a few weeks of introduction to systems concepts and an initial experience with the software system development process, I felt the students might profit from the chance to try applying these same concepts in a foreign context. There, because of the contextual freshness, the concepts might emerge with greater clarity and the students return to the traditional systems of the course with both a better understanding of systems concepts and a wider appreciation of their grander implications. I suggested the idea to the class and they decided it would be worth trying.

Experience with *Don Giovanni* strongly suggested that one opera would be just as good as another for this sort of experience. NYLOC was, that term, readying Bizet's *The Pearl Fishers* for production so an evening dress rehearsal which coincided with the systems class hour was selected for the class's "night at the opera." Each student was given the same assignment — to attend the rehearsal and to write up at least one analogy which he or she discovers between opera "implementation" and the systems work being studied in the course.

To prepare for the trip, the class was given certain written and printed material concerning the opera, was required to listen to a recording of the latter half of Act II, and was presented with a brief outline of the organizational structure of the New York Lyric Opera Company. The materials and recording were focused on a dramatic climax in Act II which involved extensive interaction of all the different personnel involved in the opera production.

PART TWO: THE NEW YORK LYRIC OPERA COMPANY AS A PROJECT TEAM

The organization of NYLOC is similar to the organization of a software project team. Figure 2 shows the organization of the production personnel as a project team. Though soloists, their respective vocal coaches, and individual singers and instrumentalists have been omitted to keep the size of the chart manageable, the main structure is clear. There is a project leader with overall responsibility (NYLOC General Director). Two sub-system leaders report to the project leader: (1) the audio systems supervisor and (2) the video systems supervisor (the conductor and director, respectively). The first has responsibility for everything the user (audience) hears, the second, for everything the user sees. Each of these sub-system leaders have both individual and lower sub-system leaders reporting to them. And, in typical project fashion, each also assumes direct management for at least one sub-system.

While several of the sub-project or sub-system personnel under one or the other of these two sub-project leaders have still other personnel reporting to them, several others do not. For example, the Group Vocal Systems Supervisor manages all the chorus personnel and is responsible for the development of the entire choral sub-system. On the other hand, the Lumination System is a one person operation. This variety in responsibilities and in numbers of upward-reporting personnel in each case is a typical project phenomena, depending on typical project realities — size of project and budget of project. In fact, the tyranny of schedules tends to make opera companies ideal models for project teams in at least one major sense — the opera team *must* come up with an implemented system *within budget* and *on time.*

Responsibilities are initially delegated to the various sub-project leaders with enough general discussion of interface

NADIR.
Ils se séparent.
Oui,
de_main
je te rever_rai
Ped

Figure 1: Sample video output and vocal source code

details to enable them all to go ahead with their individual tasks. Early rehearsals are then devoted to module or sub-system design, testing and debugging. Final rehearsals are reserved for integrating the modules and sub-systems and thus resemble full-system testing. In this respect, the implementation of the opera is a sort of bottom-up process. On the other hand, the very early rough definition of interfaces between the various sub-systems and the constant dependence on cues and stubs as modules are developed constitute a sort of top-down process. Thus, like most system projects, opera implementation involves both top-down and bottom-up approaches.

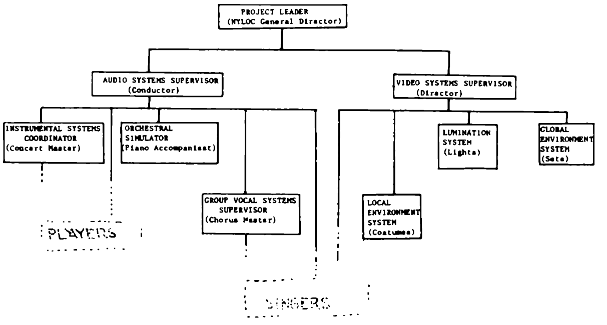


Figure 2: NYLOC Project Team

PART THREE: "DOCUMENTATION" FOR THE PEARL FISHERS

The documentation presented here is but a minute sample from the hundreds of pages existing for the *Pearl Fishers* system. For the systems analysis class, excerpts of four kinds of documentation were presented: (1) *overview* (plot summary), (2) *narrative description* (libretto), (3) *audio systems vocal source code listing* (vocal score), and (4) *user audio output sample* (phonograph recording of appropriate segments of system). Obviously no example of (4) can be included in a paper, so none will be represented or discussed here. And, though an example of (3) is shown as an opening illustration in Figure 1, this section will use the more complete *audio systems master source code listing* which includes the vocal code and much more. (The master code would have been used in class but it was unobtainable at the time.) In addition to discussing the same aspects of documentation here as in the class, this section will derive some unity from relating all examples to a single time-slice from the opera's execution.

This time-slice is underscored in the *overview* excerpt presented in Figure 3, an excerpt which adequately establishes the context for this time-slice. The systems class studied several other time-slices and their respective documentation as well, but space does not permit, nor necessity require more than this one time-slice to be examined here. The opening illustration for this article, for example, presents *video output* and an excerpt from the *audio systems vocal source code listing* relating to that time slice when the lovers embrace. It should be noted, though, that these other time slices would underscore what tremendous changes in the level of system activity occur as the act moves from the moment of the lover's embrace to a conclusion. The system must shift from an intimate, dimly-lit love duet to a furious, full-company climax with every character in the opera on stage and singing at full voice, with the orchestra playing at full strength, and with the lumination system simulating a lightning storm.

Act II *ruins of a temple*. Nurabad, the high priest, installs Leila in her position as priestess of the tribe. He tells her that she must remain in silent watch and prayer throughout the night. She is fearful of the forest sounds, but promises. Nurabad departs. As Leila trembles at the roar of wild beasts, she is suddenly reassured by the sound of a human voice. It is Nadir singing to her in the distance. She answers, and Nadir, overjoyed, tells her of his love. They embrance, but are surprised by the high priest, who has been in hiding. He calls the people together telling them that their priestess has been false to her vows. The tribesmen are ready to slay her, but Nadir shields her with his body. *Zurga, in order to protect his friend, commands the pearl fishers to disperse.* Norabad tears away Leila's veil, and Zurga then recognizes her as the same woman over whom he and Nadir had formerly quarreled. A storm arises and the people pray to the gods while the priests lead Leila away. Nadir is sentenced to death.

Figure 3: Overview documentation

The sample *video output* from a full-system test run in Figure 4 clearly shows the peak of system activity as it is being halted by Zurga's entrance. This is a critical moment for interaction of the various sub-systems. The extent to which documentation for the system details the interfacing required to implement such interaction can be determined from examining appropriate excerpts from that documentation. The appropriate segment from the *narrative description* (libretto) is shown in Figure 5. The *audio systems violin I source code* listing (concert master's part score) is shown in Figure 6. And the *audio systems master source code listing* is shown in Figure 7. Since vocal code is included in the master listing, since space is at a premium, and since the opening illustration, Figure 1, includes a typical example of vocal source code, none is presented in this section. As Figure 1 shows, no information on interfacing beyond that also carried in the master code is carried in the vocal code. Its unique component, rather than interface information generally, is the piano code which can be used during test runs to simulate or "stub" in for the orchestra.

At the point where Zurga commands a stop *(Arretez)* to the villagers' frenzied desire to slay the guilty couple, significant changes must occur.

The music must change from frenzied chorus to dramatic and isolated solo command. The activity on stage must virtually halt. The lighting and orchestra sub-systems must create a sharp change in mode. Figure 6 and Figure 7 demonstrate that the documentation carries extensive interface

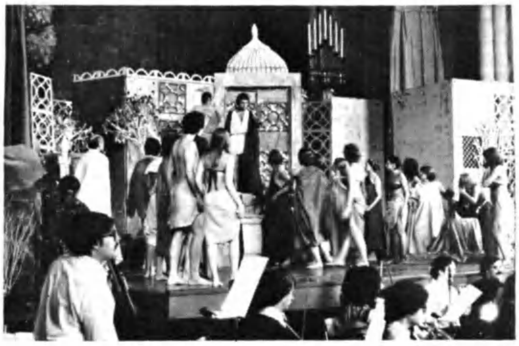


Figure 4: User video output

LEILA. Protege nous!	LEILA. Protect us!
NADIR. Venez, je vous attends!	NADIR. Come, I am waiting!
CHOEUR. Oui, pour tous deux la mort!	CHORUS. Yes, for both of them death!
ZURGA. Arretez, arretez! C'est a moi d'ordonner de leur sort.	ZURGA. Stop, stop! It is for me to command their fate.

Figure 5: Narrative Description

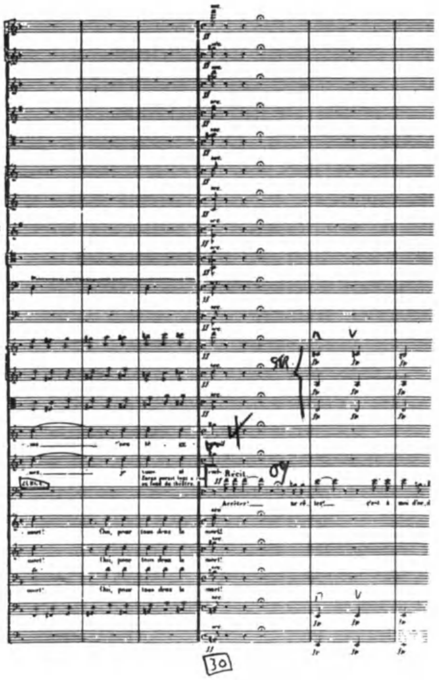


Figure 7: Audio systems master source code excerpt

detail for the audio system. The exact words of each singer, the exact pitches and rhythms of each audio system performer's notes are clearly specified. The dramatic change in mood is specified by the change in tempo at the double bar line in both source code listings' by the specification that every instrument and voice sound at full strength, once and only once, at the change of tempo; and by the specification that Zurga execute six notes in grand isolation immediately thereafter, while the instrumental sub-system remains in a wait state. The exactness of this interface between audio sub-systems is emphasized by the handwritten additions to documentation visible in both Figure 6 and Figure 7.

However, audio sub-systems interaction is only one form of interaction in the *Pearl Fishers* system. This scene certainly presupposes many decisions about interfaces between lighting, stage settings, and singer movements. Yet the documentation presented contains little or no specifications regarding such interfaces. We do find action specification stubs such as *("Surga parait tout a coup au fond du theatre")*, in Figure 7, but these are little more than can be readily inferred from the larger context provided by the singer's words. What the project team might most like to know is not even mentioned. What are the villagers to do when Zurga makes his dramatic entrance? What stage setting would maximize the impact of the entrance? Where should the guilty couple be located when the entrance begins and where when it ends? How should the lumination change during this critical moment? Not one of these questions is resolved in any way by the "Zurga appears suddenly at stage rear."

Thus formal documentation for the system carries considerable detail concerning the audio system but little concerning anything else. This is particularly noteworthy because the documentation is so extensive. The *audio systems master source code listing* alone runs to over 300 pages for the *Pearl Fishers;* the *audio systems vocal source code listing* to over 200 pages; the *narrative description* to over 30 pages; and the *audio systems instrumental source* code to over 15 volumes of 30 to 40 pages each!

PART FOUR: MAKE-SHIFT DOCUMENTATION/ A PRODUCT OF CUSTOMIZATION

In complete contrast to documentation for the audio system, that for video is make-shift and varies widely from sub-system to sub-system. Each video sub-system must go through a whole process of obtaining interface definitions; of developing tailor-made documentation concerning these definitions and anything else requiring common understanding across sub-systems; and of using the documentation to implement its sub-system (note: this documentation may not even be written down at all). Despite the differences between sub-systems, the salient elements in this process may be grasped by looking at the example of any single, particular sub-system. Such an example also illustrates why such documentation remains informal and why the collection of video sub-systems are custom components in every system which NYLOC implements. The remainder of this section is therefore devoted to the example of the lumination sub-system in the NYLOC *Pearl Fishers* implementation.

The lumination system manager had 18 lights to use. One could be used as a hand-held spot but all the others were mounted high out of reach and, once adjusted for direction, could not be re-targeted without manually accessing them from a ladder. Though the aim could not be dynamically altered during a run of the system, the brightness of each light could be, since each was connected to a separate dial on the lumination system dimmer panel.

Using this hardware, the lumination system manager based her creation of the *Pearl Fishers* lumination on four things: (1) early conversations with the video systems supervisor about his overall conceptualization of video output from the system; (2) her own recollections of work on an earlier implementation of the *Pearl Fishers* at a different site; (3) open-time opportunities to test each light in the system; and (4) observation of preliminary test runs of this version of the system.

She used (1), (2), and (4) to create a catalogue of interface points where lumination would help to define the interface both for other sub-systems and for the users. It consisted of

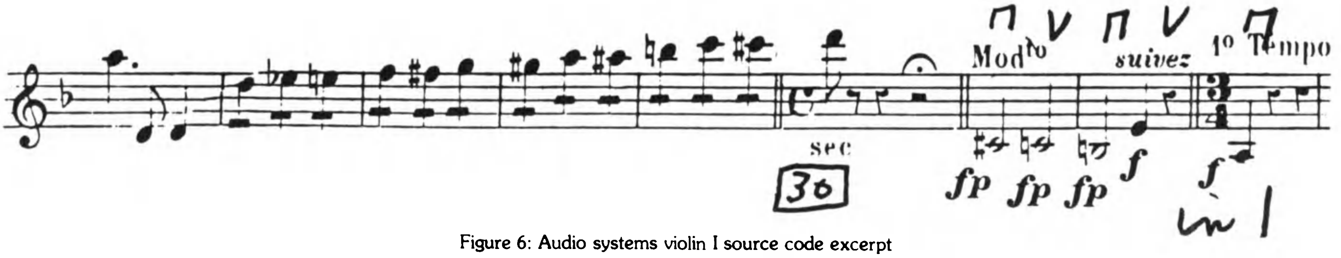
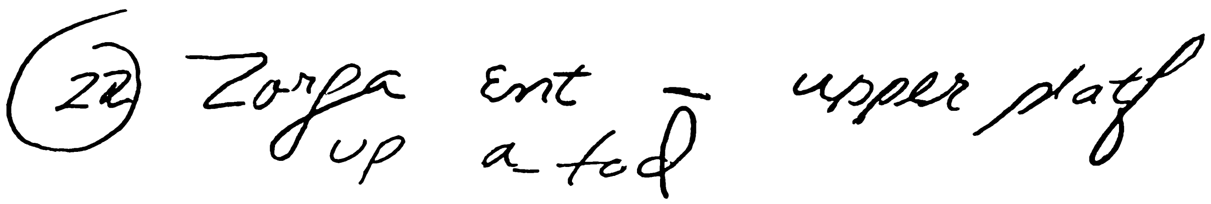


Figure 6: Audio systems violin I source code excerpt



(22) Zorga Ent — upper platf
up a foot

Figure 8: Preliminary lumination catalogue entry

34 entries describing the action occurring at the interface moment. The entry for the dramatic moment discussed in the previous section is reproduced in Figure 8.

She used (3) to number each light and create a table which would show, opposite the number for each light, the target, color, and other salient characteristics of that light. She then used this table and her catalogue during a final full-system test to create a final catalogue of working interface descriptions for lumination. As with the preliminary catalogue, this final one had 34 entries. This one, though, showed the exact lights to be used for each interface and the exact brightness for each light. Entry 22 for this final catalogue is reproduced in Figure 9. In that figure, the single digit numbers on the left are light identification, the double digit numbers on the right are brightness specifications.

It should be clear from this example just how customized this portion of the overall system is. The physical constraints alone would vary considerably from one implementation to another, rendering any but the most general interface documentation on lighting useless. Moreover, as the examples from the audio system in the previous section showed, there is no guidance as to where any of the singers should be on stage at any particular time nor any exact specification of the sets which must provide the on-stage context for these singers. Since all these would enter into decisions about which lights should be aimed where and when, no specifications of lumination could be meaningful without such details about the other video sub-systems.

PART FIVE: CONCLUSIONS

The discussion of the New York Lyric Opera Company as a project team suggested that, like software implementation project, the production of an opera involves a mixture of approaches, including both top-down and bottom-up procedures. The discussion of documentation used in NYLOC's implementation of Bizet's *Pearl Fishers* suggested that the opera implemented is a system which depends on both off-the-shelf and custom components. It also suggested that the distribution of these two component types very much followed the pattern suggested in the project analogy — off-the-shelf components went into implementing the audio system and custom components went into implementing the video system. This pattern existed because the audio system depended on standard hardware (voices and instruments capable of standard output) while the video system had to be built upon variables which changed with every implementation of the system.

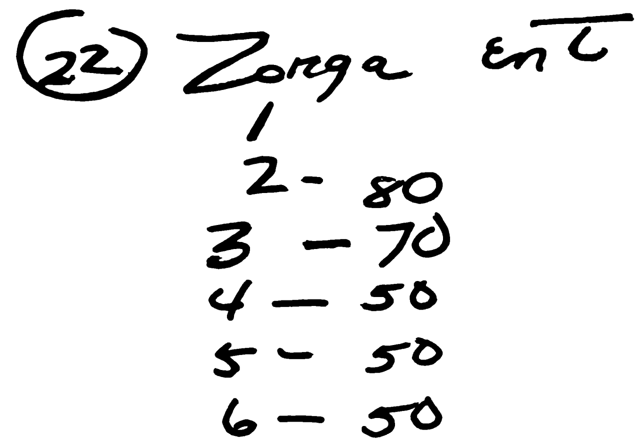
What does all this suggest about opera and computing? Certainly that the two activities have far more in common than popular stereotypes of either would imply. And this paper has examined only a few parallels — there are many more worth looking at. Long range planning, debugging, backup procedures, and iterative problem solving would all be interesting and would probably suggest further similarity between the two activities. Those who like opera and like computing may wish to explore some of these other parallels.

What about the wider implications? For those who find opera of little interest, the basic approach discussed in this paper may be applied to other human activities. For example, the systems analysis class undertook other assignments as well as the *Pearl Fishers* one. They looked for parallels to system concepts in Defoe's *Robinson Crusoe* and turned up some very interesting ones. Then they went further afield and looked for analogies in an activity of their own choice. The papers they produced in that final effort suggest the rich lode waiting to be mined. They found parallels to computing systems work in such diverse enterprises as: incubating chicks, giving birth to a child, learning to ski, playing a season of football, and preparing a family-reunion Thanksgiving dinner.

Finally, whatever else one can say about this sort of pursuit of computing's fringe benefits, three things are clear. First, successfully calling attention to parallels between computing and other significant human enterprises should weaken the popular misconception that "computer people" engage in some sort of esoteric, mechanistic enterprise, whose methodologies have no analogue in other human activities. Second, it should also demonstrate one of computing's most significant contributions to general education — it rewards the discovery of previously concealed similarities and relationships. Third, the process of looking for such parallels is both instructive and just plain *fun!*

Bibliographical notes

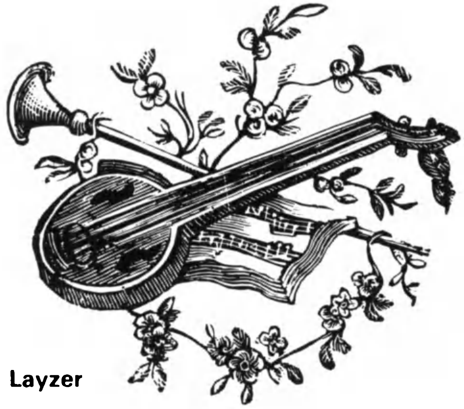
Since this system is not implemented often, documentation is sometimes difficult to locate. Since the opera is old and not frequently produced, copyrights have lapsed and pirated editions have been produced. Sample audio output is also limited. Libretti in various forms are available with recordings or in libraries. They too tend to have no coyright and in some cases no extent publisher. If you wish to see a score, look for the Choudens edition under the title *Les Pecheurs de Perles* or the Kalmus version under the title *The Pearl Fishers.* ■



(22) Zorga Ent
1
2 - 80
3 — 70
4 — 50
5 - 50
6 — 50

Figure 9: Final lumination catalogue entry

The Digital Computer: Orchestra or Composer's Assistant?



by Arthur Layzer

There are two distinct uses of the digital computer in music today: the first is to help write a score that can be played by either artificial means or by ordinary instrumental performers; the second is to actually synthesize the musical sound from a score-like specification without the intermediary of an orchestra or conventional sound studio equipment. After ten or fifteen years of exposure to these uses of the computer, most people still regard them as rather exotic.

These applications of the computer are in fact like separate magical tricks. The production of sound by the computer out of numerical specifications may be compared to a sleight of hand trick. The trick can be explained. Once it is explained there is no more magic and everything seems straightforward. The programming of a composition on the other hand is a trick of a very different kind. The difficulty is in understanding how something of aesthetic value can be created from such drab materials as algebraic transformations, random choices and Fortran do-loops. Perhaps, in fact the trick need have no rational explanation. It might be an illusion, like the Indian rope trick. After all, the aesthetic experience itself may be completely an illusion. We of course all hope that it is a benign one. If the aesthetic experience is an illusion then presumably it is enough to demand of a programmed composition that it produce the illusion in some listeners. Now, it is known that some programmed compositions have created the illusion of art among some people. The body of experience in this area is however still not large and I concede that there are serious questions of principle. To be on solid ground I shall talk mostly about the sleight of hand trick — the synthesis of musical sound by the computer.

If you analyze the production of sound by a conventional orchestral instrument like a saxophone for example, if you analyze it from a rather strange perspective you might realize that two functions are combined in one piece of hardware: the actual physical generation of sound waves by the air resonances inside the saxophone and the structuring of the air column by the holes and wall-shape of the saxophone which determines these resonances. Of course, I count the reed, the air and the saxophonist's fingers as part of this hardware.

In an electronic studio producing tape-music the situation is more or less similar from this point of view. There is a collection of oscillators with dials. The outputs of the oscillators are mixed together or serve as controlling voltages for other oscillators. The end result is a voltage output fed to the input of a tape recorder. Inside the tape recorder the electrical wave form is copied or translated into magnetic impressions on the tape. The tape is a stored record of the electrical wave-form and on playback the magnetic impressions on the tape are again rendered into electrical impulses which this time drive speakers which by their mechanical vibrations produce the final sound wave. Compared to the saxophone, there is the addition of various copying and translating stages. But just as in the saxophone the structuring that determines the musical properties of the final sound wave is embedded in the hardware, in this case the hardware of electrical circuits and the associated dials.

In the digital computer synthesis of music, the structuring inherent in the musical sound is completely separated from the hardware implementation associated with the final sound wave. The structuring is achieved by mathematically schematizing the relevant aspects of electronic oscillators and their coupling through Fortran-like programming and then outputting a sequence of numbers on a digital tape. The sequence of numbers is to be interpreted as a uniform time-sampling, at a certain rate, of the amplitude of the final sound wave, conveniently normalized. The digital tape is later scanned at a speed that corresponds to the assumed sampling rate by a standard piece of apparatus called a digital to analogue or D to A converter, which translates the sequence of numbers into a timed sequence of electrical impulses which are then fed into the input of an ordinary tape recorder. The rest of the process is then similar to the electronic studio case.

Historically, the first successful and comprehensive music synthesizing design along these lines was carried out by Max Mathews and Joan Miller at Bell Laboratories in the early 60's. This was a remarkable example of technical ingenuity that combined programming art with an appreciation of the electronic engineering aspects involved in obtaining a digitally structured counterpart to studio hardware equipment. Their first viable program was known as Music IV. A rather large number of improved modifications have appeared at various institutions across the country since that time. In addition, there has appeared a Music V version created at Bell Labs in 1968, this time by Mathews, Moore, Miller and Risset. ■

A Personal Statement

Though long interested in music and a good amateur clarinetist, I stayed away from computing for as long as possible, to my later regret. My major background is in theoretical physics and quantum mechanics, fields in which I'm still active at Stevens Institute of Technology. In 1967 I became seriously interested in computer music and audited a course at Princeton taught by Godfrey Winham. In recent years I have used the computing facilities at Bell Laboratories, Murray Hill where I am a resident visitor. Still more recently I became interested in computer animated poetry. The film "Morning Elevator" has been shown at a number of national conferences in music, literature and the arts, including the international conference on Computers in the Humanities at the University of Minnesota, July 1973. I've been indispensably aided by the programming skills of computer scientists at Bell Labs. Dr. Joan E. Miller in particular helped me write the visual program for "Morning Elevator" and in preliminary simulation with a minicomputer (DDP 24) before I ran the film off on the large GE (now Honeywell) batch system for handling movies, which used an electron-beam technique (Stromberg-Carlson) for exposing the 16mm film.

Readers of *Creative Computing* wishing to correspond further with Prof. Layzer, can write him at 161 W. 75th St., New York, N.Y. 10023.

The Transposition and Composition of Music by Computer

David B. Shmoys

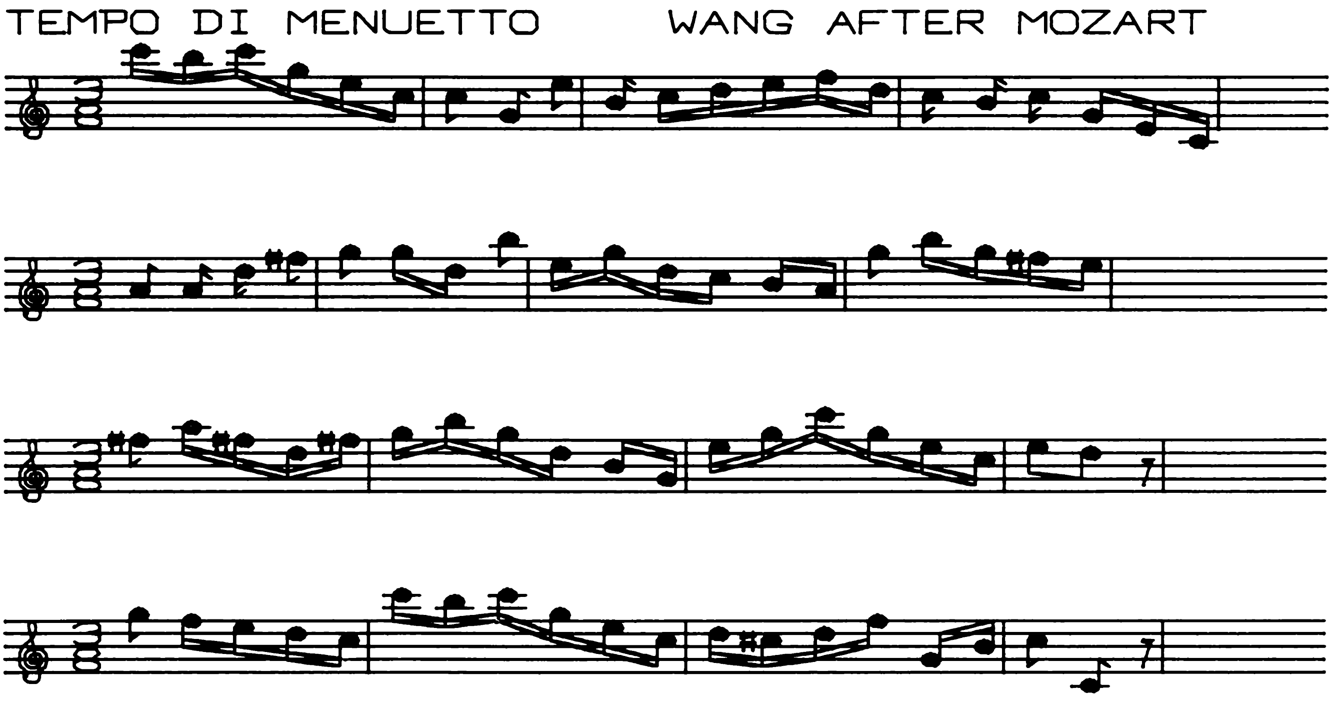
Manuscript writing and the transposition of music have long been regarded as tedious and cumbersome tasks. An efficient computer program that performs these menial chores quickly would be extremely useful to musicians. In addition, a program to get the computer to compose music that is pleasing to the ear has long been sought.

This project deals with an attempt to solve this problem using the Wang 2200 computer. I developed a program which converts alphanumber data describing musical notes into an acceptable musical score. Furthermore, I incorporated as a possible input, Mozart's "Musical Dice Game" which produces a sixteen measure minuet by random selection from a stored set of measures.

The program itself can be divided into three sections. The first section controls the drawing of the musical manuscript. The middle section transposes music any interval up or down. The final portion of the program is devoted to the composition of music using the work, "Muikalisches Wurfelspiel" written by Wolfgang Amadeus Mozart. The computer used was the Wang 2200 with 12k of core and a flatbed plotter. ■

David is now in 12th grade, in Huntington Station, New York. This project was shown at the NCC Compyouter Fair in NYC in June 1976.

David at the NCC student computer Fair.



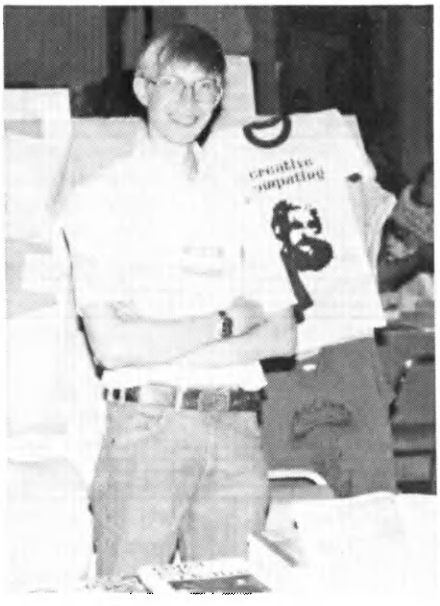
In which Publisher Dave Ahl playfully describes the genesis and on-going development of a home computer system and Contributing Editor Steve North describes succinctly, but sometimes irreverently, some of the components in this and in his system. Beware! This could happen to you!

Saga of a System

My story starts off back in June of 1975. Actually it goes back a lot further than that. Since the late 60's, I've always had a time-sharing terminal at home. But in the last few years my kids began to use the terminal for CAI, mainly drill and practice in mathematics and, of course, they like to play games. Obviously, this affected my telephone bill as well as sending the bill for time-sharing computer time through the roof. So, having seen some of the ads and articles about these new little do-it-yourself computers I decided that I would get one. In early '75 the only one that I could get any significant information on was the ALTAIR 8800 so I decided to place an order for one. I ordered it around the middle of June '75, and got it about 60 days later in early August.

Then I decided that if I built it myself and wrote up my experiences I would probably be a little bit biased. Since I certainly wanted to describe what it was like to build a computer system on the pages of *Creative Computing* I decided the best thing to do would be to find some unbiased outsider, who had experience in both writing and computer systems. So I called my good friend Steve Gray, and said, "Hey Steve, I got this neat Altair 8800 coming in a couple of weeks. How would you like to build it?" Steve, not to pass up a golden opportunity like that, said "Sure, be happy to, I've got a couple of free evenings, some week-ends. I'll build it for you." Well it probably took a little bit longer to assemble than either Steve or I had originally anticipated and compounding the task was the fact that I had ordered a fairly complete system around the basic Altair 8800: 8K of dynamic memory in the form of two 4K boards, a serial input/output interface for a terminal and another input/output interface for an audio cassette recorder as well as the CPU board itself. So that was our basic system. It took Steve many many hours to put the whole thing together. His experiences are well documented in the article that appeared in the Jan/Feb 1976 issue of *Creative Computing,* "Building an Altair 8800".

A couple of months elapsed in early 1976 when Steve's job seemed to be taking precedence over his hobby; just couldn't spend an awful lot of time on the Altair and things were sort of in limbo. Then in March I took the opportunity to go out to Albuquerque to the MITS Altair convention and I saw in person for the first time a TV Dazzler. In fact, I saw several of these rather impressive little devices. I decided that was just the thing I needed for my Altair so Harry Garland, President of Cromemco, and I had a conversation which culminated with me ordering a TV Dazzler along with a Bytesaver board to store programs for both the Dazzler and the Altair. They arrived just a few short weeks later and once again I entrusted them to my friend Steve Gray (who was probably getting a little tired of the whole thing by now). I asked Steve if he would build the Dazzler which he did (the Bytesaver I had gotten assembled for some strange reason).



Contributing Editor Steve North

We then decided it was high time to get the system to a point where it was usable. After all, I still had the time-sharing terminal at home and I was still running up my telephone bill and bills with GE time-sharing. So I said, "Hey, let's shoot for the National Computer Conference in June. Let's get the Altair-TV Dazzler set-up running with a little kaleidoscope program or some other catchy demo and show it at our NCC booth."

Well, about that time we discovered, as many other buyers of TV Dazzlers have no doubt discovered, that the TV Dazzler does not work with dynamic memory — it requires static memory. That posed a minor problem, so I took advantage of another acquaintance, Tom Kirk of the N.J. Amateur Computer Group, who happened to have a spare 4K static memory board. I borrowed it thinking, "I'll run with this one until I can get my own." Tom probably has had some second thoughts about loaning me that memory since it was not until November that he finally got it back. Which reminds me of the story: when a visitor asked Mark Twain, "Why do you have so many books stacked on the floor and all over every room of your house?", he replied "Because my friends won't lend me their bookshelves."

In any event, we didn't quite get the Altair, Dazzler, and other bits and pieces running in time for the National Computer Conference. The main reason was because of one of those "other bits and pieces," in particular a little device called a Pixieverter. Now a Pixieverter is a small enough device; it fits in the palm of your hand and only costs $8.50 from ATV research. But unfortunately Steve's Pixieverter was acting up and at that point I had not gotten one of my own.

SAGA — Continued . . .

What a Pixieverter does is convert low frequency signals put out by the Dazzler, TV camera, or other similar device to a higher frequency that can be fed into the normal antenna terminals on a TV set so that you don't need a special TV monitor or don't have to modify the TV set on the inside.

Back to NCC. What to do? The obvious solution: call upon another friend. This time Bob Radcliffe from Hoboken Computer Works. I explained, "Bob, I've got this exhibit at the National Computer Conference and I'm kind of committed to showing a jazzy hobbyist computer system (along with a Tektronix 4051 on the other side of the booth to show that we're still legitimately involved with assembled products for the education market). How about loaning me a set-up?" Well, he very kindly did that and we had a very nice exhibit with a Hoboken Computer Works machine (an IMSAI 8080 in disguise) with a TV Dazzler running kaliedoscope. It turned out to be a real big attention-getter in our booth.

However, back to our system. What was happening? Well, in fact not very much was happening and I decided, "Gee, it's been a while. I'd like to see what my computer looks like. Maybe it's about time that I take a look at it." I figured that I had a little spare time on my hands and maybe it's about time to take soldering pencil to hand. So I got the computer back from Steve along with all the other little bits and pieces. Lo and behold I discovered I really didn't have nearly as much time to work on it as I thought so I got in touch with one of our prolific contributing editors, Steve North, who had just graduated from high school a few miles away from me. I said, "Steve, how would you like a nice little summer project to get this computer system running. There really shouldn't be much to it, it's all assembled, it's all checked out, it just doesn't quite run yet. The first step is to get and build a Pixieverter of our own." No problem. Steve built it and it seemed to work

Dazzle Your TV With a TV Dazzler

If your computer is dull and boring, and only blinks its lights and types number and letters, perhaps a TV Dazzler is for you! Physically, a TV Dazzler is two boards which plug into your Altair or IMSAI, connected to each other with a ribbon cable, and which drive a TV monitor or modified TV. The TV Dazzler generates (to simplify a bit) a color picture of your computer's memory, similar to the VDM-1. But instead of generating characters, the TV Dazzler makes little colored squares appear on your TV screen. You can use it to create all kinds of interesting color graphics.

The Dazzler uses high-speed direct memory access (DMA) to read the memory of the host computer and translates the information into a TV picture while at the same time the computer is executing a program. There are two output ports on the Dazzler, and one input port. One output port tells the Dazzler whether or not to display a picture, and if so, where the starting address of the picture is in the computer's memory. The other output port is used to tell the Dazzler what resolution to use, whether to generate B&W or color; and for high resolution, what colors to use (blue, green, and red are available).

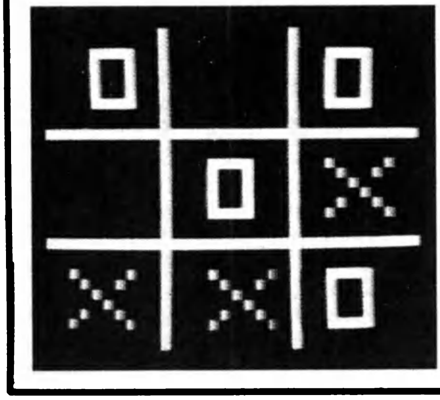
The Dazzler can generate a 32x32 element picture, which uses up 512 bytes of memory, or a 64x64 element picture, which soaks up 2K of memory. The Dazzler's memory is not on-board, so you'll have to reserve some static memory in your system for it.

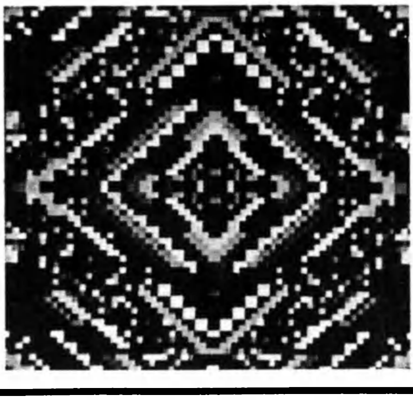
Only two bits of the input port are used. Bit 7 goes low during odd lines, and high during even lines on the TV. Bit 6 goes low for 4 msec to indicate the end of a frame.

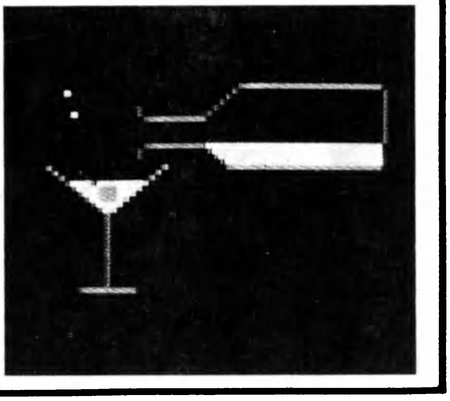
In the normal (low) resolution mode each 4-bit nybble of memory represents one square on the screen. The individual bits specify which colors and what intensity. In the high resolution mode, things get a bit trickier. Each byte represents which squares are on or off in a group of 8 squares. To decide what colors and intensity to use, the Dazzler looks at the appropriate output port. To get full color in the high resolution mode, you have to interweave frames of different colors. But enough of that here; a very comprehensive set of instructions and technical description comes with the Dazzler.

The Dazzler board itself seems of better construction than most PC boards; it uses sockets throughout and goes together very nicely as long as you don't have too heavy a hand with your soldering pencil. A simple test program is included for alignment of the Dazzler; however the alignment is somewhat tricky since all the adjustments seem interrelated! Extensive troubleshooting hints are included in the manual; at this point we haven't needed them—reliability has been excellent.

There is already a fairly extensive Dazzler software library. Kaleidescope is a Dazzler simulation of something you can get in a 5&10 for 39¢, and LIFE is a version of the popular computer game for Dazzler graphics. Dazzlemation is used to generate animated Dazzler pictures (such as an endlessly pouring Martini), and Dazzlewriter is used to write alphanumerics on the Dazzler. This program seems a little cumbersome to use, because the Dazzler doesn't have the resolution for getting many characters on a line. If you get an A/D with a joystick or two (such as Cromemco's unit) you can run things like Dazzle-Doodle, which lets you draw color pictures on the Dazzler; a chase game; and a space war game (real space war, not Star Trek) to be reviewed in an upcoming issue of *Creative*.







Processor Technology VDM-1

In any usable computer system, you need some way of getting people-oriented information out of the system. Quite often this takes the form of a hard copy device or a CRT terminal. If you have an 8080 based system with a 100 pin bus, you have a third option — a VDM. The VDM board plugs directly into the system bus, and a cable runs off to a TV monitor. Although the VDM does display characters on a TV screen, it isn't a CRT in the usual sense. In other words, you don't send out characters to it one at a time, as you do with most computer terminals. Rather, the VDM is a visible page of memory. Each byte shows up on the TV screen as a corresponding ASCII character, both upper and lower case. Since ASCII uses only seven bits (and there are eight bits in a byte), the eighth bit is used to turn the cursor on or off at a particular location.

Since you can't just send characters out to the VDM, a special routine is required to write characters out to the VDM's memory (which is on-board incidentally — the memory for the picture doesn't eat into what you already have). This routine to simulate a terminal takes up about 1/2K of memory, but it lets you do things you can't do with many CRTs. While characters are being written on the VDM, you can alter the speed of the display by typing a number from 1 to 9. 1 is ultra-slow (about 1.5 char/sec), while 9 is equivalent to about 2000 lines/minute! You can stop the VDM from displaying more characters by hitting the space bar. Subsequent depressions of the space bar cause characters to be displayed one at a time. Typing another number causes the display to continue at the desired rate. Typing control-A will turn the cursor on or off, depending on its previous state. The display scrolls, which is much more readable than starting over at the top of the screen when it's filled.

With VDM also does several handy things with its onboard hardware. An output port is used to tell the VDM how many lines to blank at the top of the screen, and the address of the on-card RAM where the display starts. There are also some DIP-switches which permit you to select: 1) black on white, or white on black picture; 2) whether the cursors blink or not; 3) whether the control characters are visible or not; and 4) whether the screen is blanked to the end of the line on a carriage return, and to the end of the page on a vertical tab. An input port on the VDM has one bit which goes low if the output port wasn't written to in the past .25 seconds.

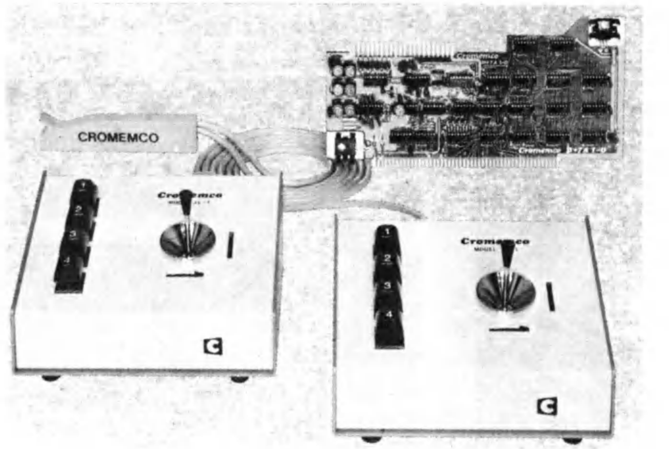
Unfortunately, there is no way to connect a keyboard to the computer through the VDM; you'll have to obtain a separate interface for that. Still the VDM is useful as the only human-readable output device in a system, and perhaps even more valuable to someone who already has a hard copy device (save time, paper, and wear-and-tear on the ears)!

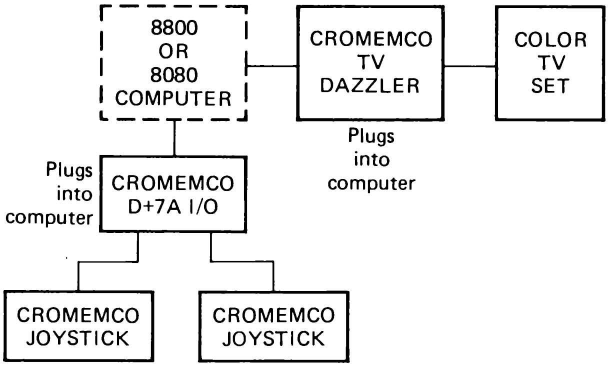
Software with the VDM includes the standard terminal replacement software, and patches for MITS BASIC. PTCo 5K BASIC and the soon to be released 8K BASIC have built-in VDM drivers. Raise a sense switch, and you have output to the standard terminal — lower it and the output goes to the VDM. There are also some VDM games we haven't had a chance to try out yet, including a realtime Star Trek game. Since all the characters on the VDM are directly accessable to the computer, it has the capability for some interesting graphics.

Cromemco A/D and Joysticks

Once you have a device like a TV Dazzler in your system, you may get tired of watching it play Kaleidescope all by itself for a few hundred hours. You'll probably want some way of playing a game *with* the Dazzler. Of course, you could use a regular keyboard for input, but joysticks lend themselves more readily to graphics, and can be used by children who haven't learned to read yet, or by people who are put off by computers. Cromemco's D+7A is a seven channel A/D; a set of matching joystick consoles is also available (you can buy one or both). The A/D itself is bipolar- that is, it accepts voltage as input in the range of -2.56 to +2.56 volts and digitizes it with sign in 40 mv increments. The A/D board also has a parallel digital I/O port.

The joystick consoles feature a joystick (of course), a speaker, and four pushbutton switches which can be read by the computer. Since two joysticks use only 4 channels, you still have 3 channels left over for other miscellaneous things. Let your computer listen and talk to the real world! Construction of both the D+7A board and joysticks are outstanding as we've come to expect from Cromemco. While an A/D setup with joysticks isn't the most essential thing you can plug in your computer, it does make your computer more fun.





SAGA — Continued . . .

fine. As a matter of fact, after a little poking around, fine tuning, putting in a capacitor here and there, and replacing one of the integrated circuit bus drivers on the CPU board, finally, the computer got working along with the TV Dazzler. Great! Story finished. Well, not quite.

Towards the end of the summer I made another mistake by going to the Personal Computing Fair in Atlantic City. I said "mistake," however, it was not a mistake going to the show. What happened was that Harry Garland from Cromemco brought along his TV Dazzler along with a couple of joysticks and the mistake was that I brought along my kids and showed it to them. After that, the two of them — that is, the Dazzler/joysticks set-up and my kids — turned out to be inseparable. All three kids just went bananas over this system. So once again I decided this is something worth having (actually, my kids decided) and once again I placed an order for an analog to digital interface (this time assembled) along with two joysticks (yes, not one but two).

They arrived in short order and I plugged them into the system. Recall, the Dazzler worked fine — it ran kaliedoscope just beautifully so one would expect that a preassembled A/D converter with two joysticks would also run fine and would produce little traces across the screen of the TV set when you moved the joystick appropriately. But instead of little one dot traces when I turned the thing on I got great big gashes of color diagonally across the whole face of the tube. It took two minute movements of the joystick to cover the entire face of the tube with color. This was not quite what I had in mind thinking back to this little electronic etch-a-sketch which my kids were playing with in Atlantic City. What was wrong? Well, I didn't know what was wrong so I called up Cromemco and said, "This thing gives me these gashes of color—what's wrong?" "Well sir, we can't really tell long distance but we can make a pretty good guess that you've got some ripple in the power supply so why don't you put a big capacitor across the power supply and see whether that works?" I said, thinking back to my days in radio/tv repair, "Does that mean a 35 microfarad capacitor or a 50 microfarad?" The telephone laughed at me and finally the engineer said, "No, it means something like, 10,000 microfarads."

Well, I rummaged around in my box of old electronic parts and found that I had a 350 microfarad unit but nothing like 10,000 microfarads. This created a small problem because 10,000 microfarads aren't available in your neighborhood Radio Shack store and anyway I wasn't absolutely sure that that was the nature of the problem.

Anyway, right around this time also I had taken my system up to the Science Fiction Conference at Great Gorge Playboy Club where it proved to be the delight of all of the people with hangovers in the morning who couldn't quite maneuver a bunny or themselves to their room the night before. A fair number of the people who looked at the TV Dazzler kind of went away with dazzles in their eyes. However, the point is that we experienced some minor problems up there due, apparently, to the fluctuating or lower line voltage. So I said to myself, "Gee, I go to a lot of different trade shows and conventions. I should really have some kind of constant voltage power supply or an auto transformer or something so that this system can be easily transportable." So I decided that perhaps a Parasitic Engineering power supply was the answer and placed an order for one. Back now to the problem of gashes of color across the face of the tube. Well, looking at what could have caused it, first of all the voltage used in they joystick mechanism is the +16 to -16 and it's regulated down to 12 volts in the joystick but, of course, for the regulators to work you'd better start off with a little bit more than 12 volts. I put a volt meter across the terminals and while just the CPU was in I got a nice 16-1/2 volts from the 16 volt power supply. Add a few more boards and it dropped down to 15-1/2 volts; more boards, 15 volts; then 14-1/2 volts; finally plugged in the two joysticks and all of a sudden the -16 volts had dropped down to -12.3 volts. Since -12.3 probably wasn't being regulated to -12 very well by the Zener diodes it was pretty apparent that that was the cause of the problem. The +16 and the two 8 volt sources were still doing fine. I determined then that it would probably take a very simple fix of a 12.6 volt filament transformer to replace the 11 volt transformer (T3) in the Altair to fix the problem along with perhaps a larger capacitor than the 500 microfarad one in the -16 volt power supply, say something like 2000 or 3000 microfarads seemed about right. Just when I had reasoned all of this out, along comes a newsletter from the Denver Amateur Computer Group with precisely the same power supply fix suggested. That is, replacing transformer T3 with a 12.6 volt filament transformer etc., etc. I could have put in this fix for about $2.69 for the filament transformer plus $1.25 for the capacitor from one of the surplus outlets but since I'd already blown $75.00 on the Parasitic

Cromemco Bytesaver

It seems to be becoming more popular to keep some programs on PROM, save thereby save yourself the agony of toggling in a bootstrap loader every time you bring up your system. The Bytesaver is a PROM board with a built-in programmer. It has room for 8K of (or 8) 2708 PROMs. Unfortunately, we can't even find 2708 PROMs in the back of *Popular Electronics*, but Cromemco sells them preprogrammed for $50. Does that give you a hint? The Bytesaver comes with a program called Bytemover on a PROM. Bytemover is used to transfer programs from RAM to ROM (for programming) and from ROM to RAM (so they can be run). By setting the sense switches, you tell Bytemover exactly what transfer to make. Obviously this gives you a very fast way to load programs into RAM (but not many of them). To program a PROM, you have to flip a switch on the Bytesaver board, and then run the Bytemover program with the proper switch settings.

We really question the worth of the Bytesaver. It does program Erasable PROMS — which are much more flexible than the non-erasable type. But these EPROMS do not appear to be widely available. Since it takes big bucks to buy PROMS for the Bytesaver, you might consider a smaller PROM board, containing at least a bootstap loader. On the other hand for "permanent" software (BASIC, a sort routine, favorite games, etc.) the Bytesaver might be just the thing.

SAGA — Continued . . .

Engineering power supply I decided I might as well go ahead with that and leave the Radio Shack/surplus vendor power supply fixes to other people who aren't quite so foolish with their hard earned money as I am. Actually I shouldn't really say that because my justification for getting the Parasitic Engineering power supply was more because it provides a good source of constant voltage under varying input conditions rather than just to fix up the gash in the TV Dazzler/joystick combination.

Meanwhile about the same time, Steve North mentioned that the system seemed to be having some CPU problems. Steve had put a capacitor across two of the terminals in the CPU to get rid of some noise in the Dazzler image. It also seemed not to be performing adds up in the high registers correctly. So I wrote MITS about the problem and returned the CPU board. Well I got back a nice little note from MITS saying that the CPU checked out just fine and when it is used with *MITS* products everything would be ok but that they could not guarantee when it was not used with *MITS* products. They also performed the clock modification at the same time which was very nice of them. Unfortunately, when I ordered the Parasitic Engineering power supply I also ordered their clock fix kit which uses a somewhat different strategy of fixing the clock than the MITS kit.

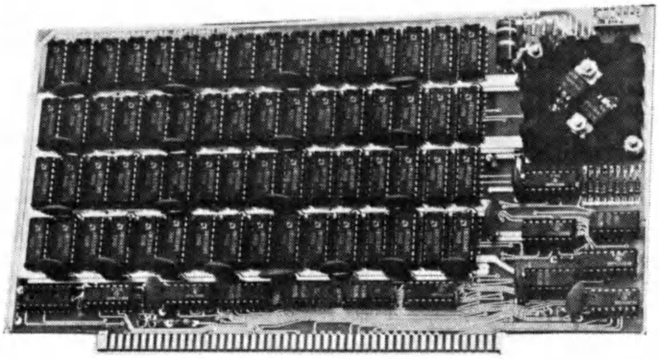
It's pretty obvious that the direction that I was going with the Altair was to make it an analog device driving a graphics terminal of some sort, say a color tv set along with they joysticks. About that time I said to myself, "what happened to the original purpose of getting a Basic speaking computer to replace the time-share terminal link up?" It wasn't hard to deduce that somehow my goals had been subverted along the way but that my original goal was still worthwhile and therefore I really should have dedicated a computer to speak Basic to my kids. Having read so many good things about other microcomputer chips such as the Motorola 6800, Zilog Z80, Intersil 6100 and so on I decided I really should have one of them. Further, having heard a lot of good things about the Southwest Technical Products 6800 kit I decided to get one. But that is a subject for another story some time.

Back to the Altair. Somewhere along the way I had to cure the problem of static vs dynamic memory. It was eventually cured by getting an 8K static memory from Processor Technology along with a video display.

Processor Technology 8K Static RAM

One static memory is as good as another, right? Well, when it comes to actually storing data, yes, provided that it is fast enough for the CPU, and doesn't get flaky, or overload your power supply (everyone's claiming low-power drain these days, but some are lower than others). The PTCo 8K does offer some extra frills though. All the ICs are socketed, a worthwhile feature. It has a dip-switch which permits you to select the starting board address in 1K increments. The 8KRA also has a plug for connection to a battery, so when you turn your computer off, the data on the board is retained. To be frank, we don't know anyone who actually uses this feature, but it seems to be the "in" thing in memories nowadays. At least it's there if you want it . . .

Assembling the 8KRA is a simple operation. You just solder IC sockets until you begin to see IC sockets even when your eyes are closed! As with all closely packed boards, you have to be careful not to make any shorts between traces while soldering.



The 8KRA also comes with two test programs. One is for a system with just a front panel and an 8KRA, while the other is for a system with a terminal. The simple program tests all the words in the first 8K memory (except for the portion of memory the program itself is in, of course), and when it finds a bad word, quits, and stuffs information pertinent to the error (such as the address of the bad word, what the program tried to put there, and what it actually read back, etc.) in the first few bytes of memory. But, surprise! The program will always tell you your memory is bad, because the first address it checks in 8192 decimal, not 8191. While there are 8192 words of memory on an 8K RAM, they start at location zero, not one. Someone goofed. Additionally, the explanation of what gets stored in the first few bytes of memory if there's an error was a little muddled. The other test program prints out a neat little picture of the ICs on the board, with X's for bad chips, and G's for good ones. Impressive! Show off to your family and friends! Despite the criticism, remember that unless you only know how to run memory checks, you're buying some well designed hardware, and not software with some minor ambiguities.

SAGA — Continued . . .

I'm not quite sure why I got the VDM other than the fact that it looked that the Altair was shaping up into something that would use a television set generally for output and it looked like the Processor Technology video display module had a lot of interesting performance characteristics.

So at this point my system looked something like this: Altair 8800 CPU; 8K Processor Technology static memory; a Bytesaver board, TV Dazzler, 7 channel analog/digital and two joysticks from Cromemco; an audio cassette interface and terminal input/output boards from MITS which at this point weren't doing much of anything, a Video Display Module from Processor Technology which also at this point was not doing much of anything, and a 4K dynamic memory from MITS (which at some place along the line I decided to upgrade with the new synchronous kit so it's 4K of updated dynamic memory which is doing absolutely nothing. The other 4K of memory was given to Tom Kirk in payment for the loan of his static memory over a rather prolonged period of time.) The output of the TV Dazzler and Video Display Module both go into the Pixieverter which is a rather inconvenient operation to switch from one to the other. I intend to try to make that a switched feature in the near future. The whole thing drives a small GE color TV set. Also I intend to hook up the cassette interface board to an audio cassette recorder and use BASIC on the system. I also intend to hook up the serial interface I/O board to some terminal as yet undetermined. (One problem of working for Digital Equipment and now for AT&T — at DEC I got hooked on a DECwriter which is a little bit out of my price league to put out for my own computer system. Now working for AT&T I'm hooked on the KSR 43 terminal which also is a little bit out of my price league. I guess I'll have to settle for something else, but at this point I haven't decided quite what.

Just a general note about trends in plug-in components of various manufacturers. Since the Altair originally came out, an entire industry has sprung up to make Altair compatible boards and even CPU's. I can't help but be very impressed with the quality of these boards that are available from these other manufacturers and now from MITS itself, particularly compared to the original boards that came with that first Altair. The Cromemco boards are absolutely beautiful, they're double thickness, all gold plated contacts. The assembly instruc-

Processor Technology 3P+S

The 3P+S is a multipurpose I/O board for use with both serial and parallel peripherals. Here's what you get:

1. Two parallel ports. These are standard TTL level ports, with data strobes (input) and acknowledge flags (output). You might use a parallel port for interfacing a keyboard or optical papertape reader.
2. One serial port. This port can be used to send serial data on either an EIA RS-232 channel (which modems and CRTs generally speak) or on a 20 ma current loop (which Teletypes usually speak). There are lots of jumpers to select options on the serial port — jumpers for UART control (word length, parity, etc.), jumpers to select baud rate, a jumper to enable or disable the current loop, and jumpers to determine which (if any) of the EIA channels you send and receive on.
3. A control port. A variety of flags can be jumpered to the control word for input to the CPU. These include flags from the parallel ports and, UART flags (such as Transmitter Buffer Empty, Received Data Available, and the error flags).

When your computer is running, and wants to input or output data, it can look at the appropriate bit on the control word and determine if a particular device has data to be read in, or is ready to accept more data. The output from the control word can be jumpered in similar ways. Some bits can be used for software control of the baud rate and UART (so that, to change from one terminal to another, you'd change the program in the computer, not the jumpers on the board. Someone we know is using a 3P+S in his Altair to run a totally programmable modem!) You can also jumper bits of the control word, both in and out, to EIA channels you're not using for serial data, which is useful if your peripheral requires signals besides "Send," "Receive," and "Ground." There's even a peripheral control driver on the 3P+S.

In all fairness, the 3P+S should be called a "2P+S+C" because it only has two parallel ports. The board seems reasonably well designed, and should be all most home computer owners need in the way of I/O interfacing, except for a mass storage interface (cassette or floppy disc). The only thing we could fault the 3P+S on is Processor Tech's documentation. For instance, the manual never explains in one place what all the little jumpers really do. The manual does explain how to connect the 3P+S to some common peripherals, but they even slip up in the instructions for creating an RS-232 interface, by neglecting to tell you to jumper the TBE and RDA outputs of the UART to the control word inputs. Test programs for the board are included in the manual, but the test program for the serial port seems excessively long to toggle in manually. It might be a lot simpler to write your own test program, and if you have difficulties try the one provided, since it does have some diagnostic features. Despite these minor inconveniences, the 3P+S is a good value, and most hobbyists would find it more useful than either a serial or parallel interface alone.

Footnote: After finishing this article, we spent torturous hours trying to get the 3P+S to talk to a working TTY. The problem turned out to be documentation. Pins J and 8 on a connector are switched in a diagram. Although the manual has been recently corrected, ours wasn't, hence 4 wasted hours.

SAGA — Continued . . .

tions with the Cromemco peripherals were outstanding in their clarity and both peripherals ran without any trouble whatsoever. I have almost the same comment for the Processor Technology boards — they are very nice boards — gold plated contacts, heavy weight board and high quality throughout. Both Cromemco and Processor Tech use sockets throughout on their boards which I personally think is very worthwhile, particularly if you, like I, have to replace an IC. In one case I had to replace the bus driver, in another the clock chip, and in still another, I had to replace a whole board full of IC's when I updated the MITS memory from the original to synchronous memory. It is just not a fun thing unsoldering IC's. Southwest Technical Products make a very convincing argument for the fact that the IC sockets are just something else to go wrong with the system — bad contacts and so on — and have convinced most of their kit builders that these sockets are not necessary. They are probably correct, but on the other hand I would probably still lean toward sockets in the event that changes might be desirable at some later date. The Parasitic Engineering components can't be directly compared to peripherals on a board since, of course, transformers, the clock IC and other parts for the Parasitic kits don't use any boards. However, the quality of the components was very good throughout. (One minor problem with the Parasitic power supply was that it assumed that the capacitor C14 in the Altair power supply was a 2000 microfarad unit or above — maybe they're thinking of late model Altairs — because mine was a 500 microfarad capacitor.)

All in all, it has been quite an experience — three people have been directly involved — myself, Steve Gray, Steve North, and many more involved on the periphery including some very, very helpful people at MITS and Cromemco who quite willingly spent time with me on the phone when I had some problems and tried to answer my questions and, in general, were very helpful. That's a very impressive part of this whole industry. Recalling my days at Digital Equipment it was very much harder then, and probably still is, on the customers of minicomputers and larger mainframes. If a customer had a technical question and was trying to modify the equipment the first thing he'd hear is "Don't." The second would be, "It can't be done." Compounding it all

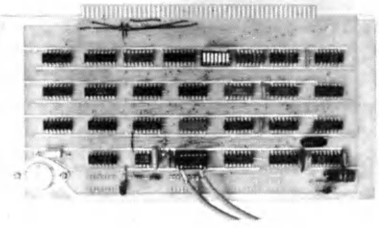
Tarbell Cassette Interface

For some time now, a cassette tape standard known as the Kansas City/Byte standard, has been in use in amateur computing. While this standard can be used with cheap recording equipment, it does have the disadvantage of being slow. With the Kansas City standard of 30 bytes/second, it takes 4 minutes to load 8K of memory. That's a long time to wait every time you bring up BASIC or some other software of that size!

The Tarbell cassette interface is also compatible with cheap recording equipment, but it can read and write data at 187 bytes/second. That means you can load 8K of memory in 43 seconds! Furthermore, if you're the adventurous type, you can try to modify the interface to run a 540 bytes/second, although you might find your bargain basement cassette recorder isn't quite up to the job. Tarbell is still recommending exchange of data at the 187 bytes/second speed.

The kit itself is fairly simple to assemble, it's just a PC board with 30 or so ICs, some discrete components, and a few jumper wires. The cables that are used to connect the interface to a recorder are also included. To adjust the interface, you just plug it into the computer, hook up the cassette recorder, turn everything on, and play a test tape consisting of nothing but sync bytes (E6 hex). You then twiddle a trimpot until a LED on the board comes on, indicating that sync bytes are being received. That's it.

The manual that comes with the kit includes instructions, recommendations on what kind of recorder to use, operating instructions, bootstrap and output programs with checksum, an example of how to use the interface under control of a BASIC program, instructions about how to modify the interface for computer-controlled start-stop, and patches for CLOAD and CSAVE in BASIC. Tarbell also markets two software packages: a chess program we weren't able to review, and a modification of Processor Tech's Software Package #1, an assembler/text editor/monitor. Tarbell's modifications permit you to write both source and object code out to the interface, and then read the code back in or check what you wrote for validity (so you won't save a big program and discover that the recording was bad and you've lost the file).



A tape of the software is $5 and a manual with a source listing is $5.

Obviously having a cassette interface that runs that fast and isn't expensive (depending on your point of view) is nice, but there are a few problems. First of all, there have been some complaints of problems with getting the interface to work. You're not really taking a risk, though, since Tarbell will repair the board free of charge, or refund your money, within 90 days. Another thing — you can't just write characters or code out to tape, and then read them back in. Whenever you write out a block of data, you have to first send out a start byte, then a sync byte (E6 hex), and then all the data. You don't have to bother with these control bytes when reading data back in — just reset the interface by sending a 10 hex out to the control port, and start reading. Also, since the interface is a synchronous device, with data being sent out in blocks, you can't do too much processing in between writing characters out- under 5300 instructions. Under most circumstances that wouldn't be a problem anyway; and if you do have to do a lot of processing, as in BASIC, you could always put all the data you want to record in a buffer and when you're done processing, write all the data out to the cassette at once.

The last problem is that there really is no cassette tape standard, with people using the MITS cassette interface, *Byte* standard, and Tarbell. But most software is available on Intel format papertape. If you have a friend with an 8080 based system and a papertape reader, you could always try to get him to copy whatever software you buy to a Tarbell cassette tape. Actually, the computer store I frequent (Computer Mart of NJ) is willing to sell you software on one of the usual mediums, and then copy it onto your own cassette.

SAGA — Continued . . .

was the fact that he probably couldn't reach the right person to talk to anyway. About the best you could do generally was to talk with a sympathetic field service technician who may not always be as knowledgeable as the customer himself. At DEC, we certainly had a number of customers that fell into that category. The general feeling was, "Oh, you'll void the guarantee," or "You'll screw up the system somehow and we can't be responsible and stand behind it if you do that." Well that may be quite true for the mini and mainframe manufacturers. But I think it's heartening that people working for the manufacturers of the microcomputer kits and peripherals and virtually everyone that I've come in contact with in this fledgling hobbyist industry is openly helpful. Although you occasionally get notes back as I did from MITS saying that it works fine with MITS products and so on, for the most part people are thoughtful and recognize that you're not going to use all the products from just one manufacturer in assembling a system — that's just not the nature of the hobbyist computer user. (I use the term "hobbyist" here very loosely because I'm including anyone who is building a microprocessor unit whether he's building it for a small business, for a school or just for fun at home).

If there's one area that the hobbyist computer manufacturers fall a bit short in it would have to be the area of documentation. The supporting documentation and checkout programs are, in many cases, very weak. In fact there is nothing with the Altair that I would put in the category of a diagnostic program; for example, something that checks every single bit of memory, turning it on and off, and then giving a check sum or diagnostic at the end. At Digital Equipment, field service technicians and software support people were armed with a case full of diagnostic and checkout programs for every single component in the system. This is severely lacking in many (most) of the hobbyist computer kits or peripherals I've seen to date. Consequently at this point there's a great necessity for clubs — the Amateur Computer Group of New Jersey of which I'm a member — or any one of the other several hundred computer clubs across the country where hobbyists can learn from the experience of one another what works, what doesn't work, what's likely to go wrong. Good old-fashioned trouble shooting techniques are also very practical. Fortunately a computer is a very logical device by nature and, in fact, is fairly easy to diagnose by simply tracing back through the circuitry and identifying what component or components are likely to be at the cause. (That's how I identified the bus driver IC that was malfunctioning in the Altair. It was simply a matter of going from one schematic diagram to another and tracing all of the components that had some influence on the signal, then trying combinations and seeing which things worked and which things didn't. Finally it narrowed down to just one component. Of course, it was not the entire bus driver that had burned up but just one of the gates).

This computer system which seems never to be quite finished is probably quite typical of most hobbyist/school microcomputer kits. Speaking for myself (and not necessarily for Steve Gray or Steve North, who actually had the burden of construction on their shoulders) it's been a real ball. And in the future, even though the self-contained black boxes are on the way, I look for increased fun as well as challenges from my system(s). ■

Parasitic Engineering

Parasitic Engineering sells two fix-kits designed to alleviate bugs in the original Altair 8800. One of these is a power supply kit, which uses a constant voltage transformer and high-current rectifiers. Unlike a linear transformer, a constant-voltage transformer, with the help of a special winding and a capacitor, maintains a constant voltage at the secondaries while the primary voltage can range from 90 VAC to as high as 140 VAC. To install a Parasitic power supply, you have to remove the back panel, and disconnect the power supply wiring. After replacing the original Altair P/S transformers with the big fat C-V transformer and capacitor, you have to do some minor surgery on the power supply board to install the high current bridge rectifiers. All that remains then is to reconnect all the power supply wiring. We found it necessary to install a Parasitic Power Supply in our Altair when we found that it didn't have enough muscle on the -16v bus with a full card cage, particularly with dips in the line voltage. The Parasitic Power Supply is rated at 12 amps at 8 volts, and 2 amps at ±16v. Not quite an IMSAI power supply, but adequate! (Note that the writer enjoys taking jabs at the Publisher's Altair.)

Parasitic also sells an Altair clock fix kit, which promises to do everything but debug your programs for $15. Since some early Altairs seem susceptible to clock sync problems, you might find it worthwhile. De-installation of the existing clock IC on the CPU board, installation of a 94618 in its place, as well as some cutting of old traces and replacement of a handful of resistors and capacitors takes an hour or less to do.

What we really want to know is, why did they name their company "Parasitic Engineering?"

Addresses

Here are the names and addresses of the manufacturers mentioned in the above article.

MITS, Inc.
2450 Alamo S.E.
Albuquerque, NM 87106
(505) 243-7821

Cromemco
2432 Charleston Road
Mountain View, CA 94043
(415) 964-7400

Processor Technology Corp.
6200 Hollis Street
Emeryville, CA 94608
(415) 652-8080

Tarbell Electronics
20620 South Leapwood Ave., Suite P.
Carson, CA 90746
(213) 538-4251

Parasitic Engineering
P.O. Box 6314
Albany, CA 94706

ATV Research
13th and Broadway
Dakota City, NB 68731
(402) 987-3771

Computer Mart of New Jersey
501 Route 27
Iselin, NJ 08830
(201) 283-0600

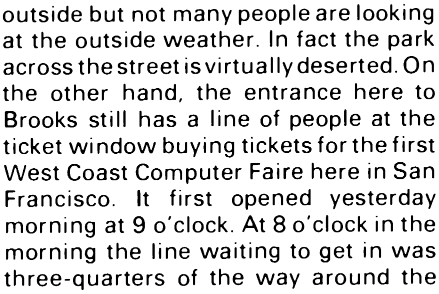
In which our publisher dons his reporter's hat and wanders around the Faire talking briefly to various and sundry people. Here it is: the color, the noise and the pagentry—unabridged and practically unedited! Are we for real? Judge for yourself.

The First West Coast Computer Faire

David H. Ahl

It's 3:30 Sunday afternoon April 17, 1977 and I'm standing here at the main entrance to Brooks Auditorium. Absolutely gorgeous sunny blue-sky day outside but not many people are looking at the outside weather. In fact the park across the street is virtually deserted. On the other hand, the entrance here to Brooks still has a line of people at the ticket window buying tickets for the first West Coast Computer Faire here in San Francisco. It first opened yesterday morning at 9 o'clock. At 8 o'clock in the morning the line waiting to get in was three-quarters of the way around the block. As of noon on Saturday 8,000 people had been admitted. Badges and programs had virtually run out and there was a momentary crisis. A new supply of both were obtained, the lines resumed inching forward, and the crowds continued well into the afternoon.

Today there were probably another 3,000 to 4,000 that weren't in the Faire the first day, thus bringing the total attendance to this first West Coast Computer Faire to 12,000-plus. (Official attendance was announced at 12,657.)



Peace and quiet Friday before the show opened, and time to finish a few signs.

As I walk into the main exhibit hall, both eyes and ears are assaulted with a variety of sounds. A number of music and speech synthesizers are all playing and talking at once. In addition, many people have displays using Advent projection-tv devices so that there's a virtual kaleidoscope of colors and sounds throughout the exhibit hall. This late in the day it's still difficult to move up and down the aisles. The aisles are crammed wall to wall, with four to six people deep around each booth. I found earlier it took an average of 15 minutes to get into a typical booth to talk to anybody in depth. There are approximately 140 booths in the main convention area and another 30 booths, actually mini-booths, around the outside of the main room—175 exhibitors in all.

I've just spotted Jim Warren, organizer of the whole show. In your own words Jim, how would you describe it?

Warren: A mob scene.

Ahl: A little overwhelming. What's the estimated attendance?

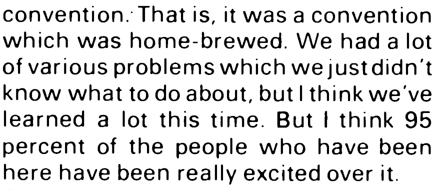
Warren: Yesterday evening we had in excess of 8,000 people and we've probably gotten another couple of thousand today. There's been a fairly steady stream through the ticket booths today so I would guess that we're pretty close to 10,000 people which is really pretty much what we expected. [As noted earlier, actual attendance was over 12,000.] But 10,000 as a number is very, very different than 10,000 walking, talking computer freaks. That's for sure. This was a mob scene. I think 95 percent of the people were just excited and really had a good time and it has really been a good day. It's also very definitely been actively characterized as a home-brew convention. That is, it was a convention which was home-brewed. We had a lot of various problems which we just didn't know what to do about, but I think we've learned a lot this time. But I think 95 percent of the people who have been here have been really excited over it.

Ahl: Are the exhibitors pleased?

Warren: As far as I can tell, almost everybody is really pleased. With the imperfections in the operation that we've had, people have nonetheless been patient and calm and seemed to enjoy what they were doing and even enjoying standing in line coming in just reading the programs and reading a copy of the *Silicon Gulch Gazette*.

Ahl: Are you planning to make it an annual event?

Warren: I don't know. Probably so. It would be terribly wasteful for us not to do it at least one more time because there's been such a tremendous learning process this time. For five months we



"One game of Tank War coming right up." Cromemco's Z-2 system was a major attraction at the Creative Computing booth

creative computing
#1
in software and applications
CREATIVE COMPUTING MAGAZINE
creative computing magazine

Continuous mobs during the two days of the show. Exhausting, but also exhilerating!

have been feeling, "By God I can't get enough help." You know, there's just too much to do. And there's been a tremendous learning period. We've learned so much now as to how to put on a good show that it would be very foolish of us to not do it again. It would be very wasteful.

Ahl: Can you afford the time to do it again?

Warren: With a year's lead time, which we would have, possibly so.

Ahl: As publisher of *Dr.Dobbs,* how do you justify so much time for the Faire?

Warren: *Dr. Dobbs* has always been a part-time activity. I'm a computer consultant; I have been for about ten years, and it just so happens that the last five months I've been consulting for the Computer Faire, which happened to be my own company.

Ahl: Will you make enough money on this to justify the last few months?

Warren: I had a pretty low salary, based on an hourly rate. But the point of doing this wasn't really to make money. I mean, back in the 60's we had happenings in San Francisco, and San Francisco was meant for that, and this is just another variation on it except it's a decade later. Back then it was power to the people and now it's computer power to the people.

Ahl: Thanks, Jim.

As I walk through again to the exhibit hall, my eyes and ears are assaulted with sounds and sights. And people! It's wild! Here at the beginning of the 100 aisle on my left is the two-booth display of Polymorphics Systems—a beautiful professional display. In contrast to that, right across the aisle, Component Sales, Inc. has a booth with polybags of all types of components, keyboards, chips, and even tape drives out on the counter. People are clustered around three deep and buying up everything they can get their hands on. Next to Polymorphic is a booth from Teletype Corp. which seems to indicate that even some of the large and well-established companies are beginning to recognize the hobbyist computer market as a viable market. Teletype is showing the printer from Dataspeed 40 as well as the new KSR 43 terminal and the old standby ASR 33. As I proceed down the aisle there is a small one-board computer system from Western Data Systems. Across the aisle is a display from TSC Software showing tapes of compilers, interpreters, and game packages for the 8080 and 6800 systems. Further down the aisle is Northstar Computers with their floppy-disc system, across from a display of the American Radio Relay League. Somewhat surprising a little further on—*Datamation* magazine. So not only are large manufacturers such as Teletype Corporation represented, but large industry publications like *Datamation, Computer Decisions, Mini-Micro Systems* are here too.

In the corner at the end of the first aisle is a large booth put up by the Byte Shops—with an assortment of books, computer systems, and peripherals. In addition, many of the individual Byte Shops and other computer stores in the area have booths throughout the hall. Then another large booth of Jade Co. doing a landslide business selling electronic components. Looking up in their booth they have a big sign, hand-lettered, "Sunday Show Special - 10% off on every item on the table." 2708 PROMS turn out to be a big item and many companies are selling them here at the appropriate price of $27.08. Around the corner is a booth for E&L Instruments. Across the aisle from them is a display called the Body Microcomputer, which is actually a long T-shirt being nicely shown off by some rather shapely girls in front of the booth, modeling it.

Going down the 200 aisle we see booths from Motorola Semiconductor showing their popular 6800 chip and associated drivers and circuits. Next is a Japanese display: Sord Microcomputer Systems, who appear to have a complete microcomputer system with floppy disc and terminals. IEEE *Computer* Magazine has the corner display diagonally across from our own *Creative Computing* display. People at this point—it's around four o'clock Sunday afternoon—are still three deep around the *Creative Computing* display. There's a mob of young ten-to-twelve-year-old kids gathered around the Cromemco Z-2 computer system running a new game called "Tank War"—two tanks maneuvering around on the screen, firing at each other with appropriate sound effects as the bullets hit the other tank. Some of them (kids, that is) have been there all afternoon and still show no sign of losing interest. Cromemco very kindly loaned us a Z-2 system with TV Dazzler and two joy-sticks. We've been alternating between Tank War, Chase, and Space War all weekend the system has been completely reliable.

Here's Howard Fullmer and Gene Nardi of Parasitic Engineering. (Parasitic is showing their Equinox 100 computer system for the first time here.)

Ahl: What do you think of the show?

Fullmer: It's incredible, all the people we've had here.

Ahl: Every time I've been past your booth there have been at least five people around it and I've had a heck of a time getting in to see your system.

Fullmer: We really haven't had that much time to observe because we've spent all of our time over there talking to people.

Ahl: What kind of people are you finding are interested in your system or what kind of people are you meeting here in general?

Fullmer: Well, of course, all the dealers are interested in our new system. but also, we've met a lot of people who have never thought about computers and just came by to see them. They show a lot of interest in how to program one of these—what does all this software mean and what do I need to get going? I want to write a simple inventory package for my business—how do I go about doing that? So, small-business people, hobbyists and a lot of curious novices. But just about everyone here is interested in a personal computer, deeply interested.

ional Semiconduct

Big companies were well represented at the Faire.

Ahl: Are people going to buy this year or consider?

Fullmer: I think this is a faire of people who want to do something now—they're not all waiting for the $500 Commodore. They're a lot more together than that. They understand that there are a whole lot of different ways to go and they're looking at all of the ways.

Ahl: Are you having a good reception to your new system?

Fullmer: Oh absolutely.

Ahl: Do you think you're really going to get off the ground?

Fullmer: Yes. The kind of comments we've overheard are people saying, "You've got to get over and see that one!"

Ahl: So you think it's going to be a different kind of magnitude than selling power supplies and clock-fix kits.

Nardi: Yes, definitely!

Ahl: Can your production line keep up?

Nardi: Ask me after sx months, but we obviously think we can or we wouldn't be doing it.

Fullmer: We've built the company from literally nothing and it's been just fantastic.

Ahl: You're going to continue to call it Parasitic Engineering, right? Even though you're not parasites anymore.

Fullmer: Well, we thought we'd call it the Equinox 100 by Parasitic Engineering so we can go either way.

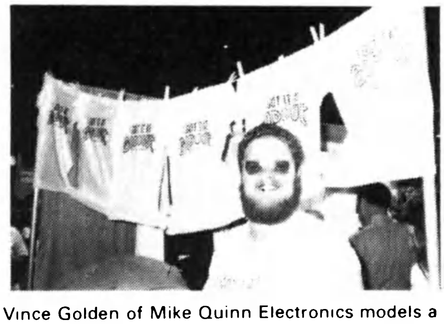
Ahl: Thanks, and good luck! Turning around who do I see but Ted Nelson. Ted, what do you have to say about this Computer Faire?

Nelson: There seems to be a lot of it and what I've seen has been choice. I think we've seen here that the computer world has suddenly broken in two. There's the straights and a strange coalition of hobbyists and the most technically competent and ambitious people throughout the field. A remarkable coalition that is going to do remarkable things.

Ahl: What do you think of the speech and music systems around?

Nelson: Just lovely! By the way, when's my comic strip coming out?

Ahl: Next issue, May-June. Our printer is on strike but we have another one and hope to have copies for the Trenton Fair. Turning around, who do I see, but Lou Frenzel of Heath. Lou, I see by your teaser ad you're announcing a couple of computers and several peripherals.



Vince Golden of Mike Quinn Electronics models a Godbout T-shirt.

Frenzel: Actually three computers, depending on how you classify them. Two of them are definitely full-blown general-purpose computers; the other is not. The line of peripherals will be quite complete—naturally there'll be some missing gaps in the beginning but you can't do everything at once.

Ahl: Will any of them be offered assembled?

Frenzel: No. Trying to get a wired product approved internally is close to impossible. Not that it can't be done, but we're a kit company and that's what we do best.

Ahl: We're all looking forward to seeing your products too.

As I continue down the aisle to my right is the Midwest Scientific Instruments booth. Jim Warren has just announced over the loudspeaker system that the top-dollar-value door prize, a floppy disc of Midwest Scientific, was just won so there's quite a crowd around the booth waiting to see the winner show up. Across the aisle is a display by Ohio Scientific Instruments with a beautiful color TV display, the OSI Challenger and floppy-disc system along with several other systems, all running and looking quite handsome. As I continue further down the aisle—I'm having a little trouble getting through—there's a huge crowd gathered around the Digital Group display on one side of the aisle and across the aisle Southwest Technical Products with an equally large crowd of people gathered around. As we continue on into the next aisle we find Apple Computer, once again with a beautiful display: an Advent projection TV device showing color graphics. Apple is demonstrating for the first time at this show the Apple II computer system. Here's Mike Markkula, Vice President of Marketing of Apple Computer.

Ahl: Are you pleased with the attendance here?

Markkula: Yes.

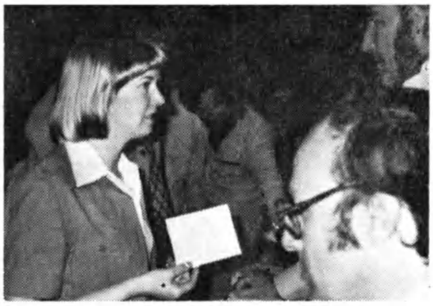
Ahl: Wildly or just mildly?

Markkula: In terms of numbers—wildly, but in terms of the reasons for their attendance, I'm not sure exactly why so many people are here. An awful lot of them are just plain curious as to what's going on. I'm surprised that a lot of them spent the price of admission, but I think they've all enjoyed it. I expected a few more people on a higher knowledge level. I'm very surprised at the whole thing.

Ahl: I would guess the Apple II would be suited to the kind of people that came to the Faire.

Markkula: Absolutely.

Ahl: You've got essentially what you call a black-box computer. Self-contained. You don't really have to know anything except how to hit a switch. What type of market are you aiming at? Somebody that wants to do programming, play games or what? Anything specific?



Constand crowds jammed the TDL booth to get a glimpse of newly revealed Xitan system. Donna Galleti of TDL talks with visitors to the booth.

Markkula: All of the above and more. We really want to be *the* computer company, not the small-business computer company or something else—just the personal computer company! So that's the reason you see a molded plastic case, BASIC in ROM, and so on. In fact we want to extend the whole concept to make it even easier to use.

Ahl: I noticed that your demos use various types and even color graphics. Is that all in your BASIC?

Markkula: It's all in the BASIC. We have COLOR EQUALS, PLOT a horizontal line, PLOT a vertical line, PLOT a point, etc.

Ahl: That's some BASIC! What size is it?

Markkula: 6K. It's an interpreter. It's also got a feature that nobody else has. We can have variable names any length up to 256 characters.

Ahl: I assume it has string manipulation and functions?

Markkula: Yes, string functions and matrix manipulation.

Ahl: What's the MPU chip in the Apple II?

Markkula: The 6502. It's the most efficient chip for what we're trying to do.

Ahl: What would a complete Apple II system require in terms of memory for the beginner?

Markkula: 4K is more than adequate. Remember the 4K that comes with a standard minimum system is all user space because the BASIC is in ROM. So almost all of the 4K RAM is available for programming and data.

Ahl: What would a system of that configuration cost?

Markkula: $1,298.

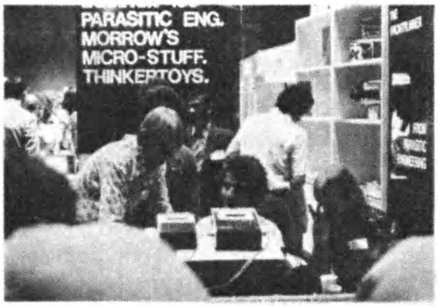
Ahl: Is that assembled?

Markkula: Assembled, tested complete with two game paddles and a complete carrying case so you can carry it around—all the cords and manuals and operating information.

Ahl: The "paddles" are those things that look like joysticks?

Markkula: Yes. You can hook up four paddles or two joysticks and pushbutton inputs. You can have all kinds of output, to a speaker for example. In fact, there's a speaker already on the board, although you can add four more in each of the paddle boards.

Ahl: When do you start making deliveries?


PARASITIC ENG.
MORROW'S
MICRO-STUFF.
THINKERTOYS.

Howard Fullmer (left) and Gary Fitz (center) of Parasitic Engineering demonstrate the Equinox 100 computer.

Markkula: The end of May.

Ahl: Thanks, Mike.

Continuing down the third aisle we find a huge triple booth of *Byte* magazine. Behind the table Virginia Peschke, the publisher, acting in the role of a salesperson.

Ahl: Virginia, what do you think of the show?

Peschke: I think the show is just great—marvelous. It's dynamite. And my feet hurt.

Ahl: I know what you mean!

It's very late in the life of the show—4:30 on Sunday afternoon and the show closes in just half an hour—and yet people are still three and four deep around every demonstration machine at the Processor Technology booth. The SOL computer is generating a lot of interest; one of them is hooked up to a music generator which is making very interesting sounds and generating a lot of enthusiasm among people attending the show.

Walking by the Prime Radix Booth, one's ears pick up the sounds of Handel and Bach coming out of a six-channel music synthesizer board just introduced by Prime Radix.

Over in the next aisle we have People's Computer Company and Dymax next to each other. Right next to them is Computalker Consultants who have a speech synthesizer playing a rather clever saying—"Hello, I'm Computalker, the speech synthesizer designed to plug into the standard S-100 bus on your 8080 microcomputer." The Computalker allows a variation in the speed without altering the pitch of the spoken word. A very interesting concept.

Next is the Parasitic Engineering booth with the Equinox 100 Computer System—those were the people we were chatting with earlier of Parasitic—and people still at this point are gathered four deep around that new computer system.

Continuing on into the next aisle we have Cromemco running very much the same games that we were running on our Z-2 system at the *Creative Computing* booth—tank war, space war and a chase game. Once again an army of young children playing these games on the dual joysticks. Continuing down the aisle we have a new company called Smoke Signal Broadcasting which has SWTPC 6800 compatible peripherals. Across the aisle ICOM with their mini floppy discs and right next to them Micro Designs with a cassette recorder. A little bit further we have Galaxy Gystems with still another kind of music synthesizer. Across the aisle from them, Godbout Electronics with various types of components and S-100 bus peripherals. Next to Godbout is National Semiconductor with beautiful chrome fixtures and blue carpeted booth—once again demonstrating the commitment to this hobby by some of the very large manufacturers.

A few moments ago I met Bob Davis. Bob is with Intel, Microcomputer division. His job is "manager of hobby marketing and personal computer products." So, a company like Intel, the leading chip maker, has a manager for the personal computing movement! Aha!

At the end of the aisle is a company called Heuristics Inc. with a new product called the Speech Lab. The Speech Lab is the first peripheral that permits a computer to recognize speech (for under $300!).

Continuing on down to the next aisle we have a huge crowd gathered around the Mr. Calculator booth. This is the place where the rumored Commodore system is being shown. The $500 computer with 12K memory, cassette and TV interface, and keyboard is self-contained in a small cabinet. If the $500 price comes true it would certainly be a revolutionary development in the whole field. Approaching me now is Lou Fields, President of the Southern California Computer Society.

Ahl: What do you think of the show?

Fields: Well, I didn't have enought time to see everything that I wanted to see but I thought it was very good.

Ahl: Did you attend the sessions?

Fields: I attended some sessions. But in terms of the sessions I really think the stimulation is getting to talk to people individually. You have the people around here who designed the equipment and who are the creators in this field and they're here at the booths. I think that's enormously valuable in terms of the catalyst of this whole thing.

Ahl: I assume that SCCS has pretty good representation here.

Fields: Yes, I've seen a lot of our members.

Ahl: How many members are there currently?

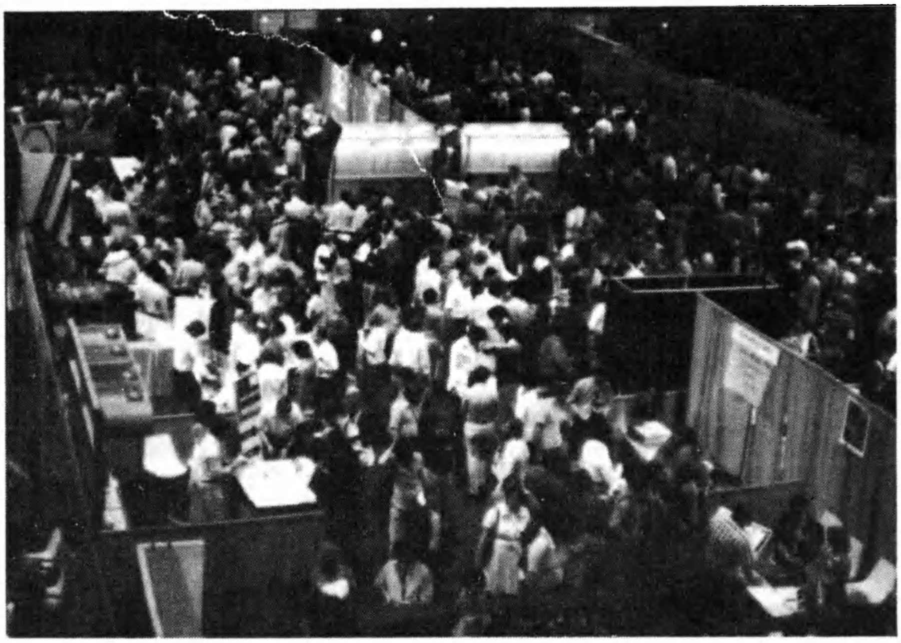
Fields: We have about 8,000 members around the world. We have chapters in firms and communities in California and now many communities as well around the world including Mexico City and Tokyo. I think small-business applications are going to be even more important out of the country then they are within the country.

Ahl: You spoke of the members outside of California—how about the members in the Southern California area where it started? How many—what percntage of the membership is there?

Fields: I'm not sure of the absolute numbers but I guess off hand that it's probably about somewhere between 2,000 and 3,000.

Ahl: So of the 8,000 total, at least 5,000 are scattered elsewhere. That's very interesting.

Fields: Yes it is, and of course our name is very misleading and we've been considering something that's been more accurate. We did a poll this month to see



View of the West Coast Computer Faire from above

After the Faire, the breathtaking California coast provided a welcome change of pace, particularly for the out-of-state visitors.

what the members felt about that. Many of them feel we should change the name to something more appropriate, and I think we will.

Ahl: I would think that many members outside of Southern California might be pleased to be associated with the SCCS name.

Fields: Well, some of them are. Some of them find it charming and quaint but a lot of others are very annoyed by it and they have negative feelings that "Oh, Southern California is a junk society and we don't want to be involved with that." So there are two sides to the coin—there is the glamour and the fact that the bulk of the small computer industry is located in Southern California.

But I think if you want to know where the most important area is in the country for members and microcomputer activities I'd say Philadelphia. It's not in California—it's in Philadelphia. Draw a fairly small radius around the eastern states and you get two-thirds of the population of the United States. So on that basis, Southern California is insignificant.

Ahl: However, we're probably not looking at the whole population - we're really looking at a demographic subsegment.

Fields: OK - so there are a lot of ways of looking at it. The people active in the field today are certainly clustered in California and half a dozen other places. But in terms of the real bulk of the people who are *ever* going to be active in this field, it is your area (East Coast) that I feel has the most potential. You've got the bulk of the universities, the bulk of the manufacturing companies of all kinds, and the bulk of the people.

Ahl: Well, you're right. As the population at large gets drawn into the movement, yes, it's certainly going to shift to the population centers.

Fields: That's inevitable. It will. You have something developing actively in a given area because of a few individuals, really. It developed in the Los Angeles area initially because of the interest and actions of Don Tarbell and that's where it started. Do you know the history of the SCCS?

Ahl: No. Please go on.

Fields: Well, Don Tarbell was an engineer at Hughes who was into hardware and software design. He went to the MITS caravan and got an idea for this audio-cassette interface board. he designed that and then went to sell it—not as a board but as a set of plans. He put a small index card up in one of the local electronics stores offering the plans for $2. I walked in and said, "Gee, that sounds interesting." So we met at his house with a bunch of other people and that's how it all started. It really started with the judge of the superior court of the local area who was a ham and went on from there.

Ahl: Very interesting.

Fields: It's an individual thing. I don't think things happened by committees anyway. Like yourself—you created this fantastic magazine and you've done an incredible thing. I'm sure you haven't done it because of 40 other people who wanted to get into it. It was your own inspiration, it was your own genius and knowledge that put it together. And that's the way the whole industry is going.

Ahl: Thanks, Lou.

I had hoped to get Sandy on here but things are still hectic at our booth. Sandy, my wife, was helping out here at the booth along with a number of college students I had recruited via the timeshare computer network at the University of San Francisco. While they use the timesharing system, most of the students there had very little idea what pesonal or hobbyist computers were all about and coming to the show was a real eye-opener for them.

One interesting observation Sandy made is that most of the people involved in this industry are young late twenties or early thirties, from programmers and designers to presidents and founders of companies. We're dealing with an industry that appeals to the young at heart. Even more, it inspires the entrepreneurial spirit in people. This is something that has been dead, in my opinion, for a long time in the United States. Many college students today are looking for a secure job with an established company; they're not seeking a position where they have a lot of risks or where they're putting their own judgments on the line. Yet in this new hobbyist, personal-computer field, it's innovation, creativity and individuality that really count. These personal qualities are rewarded in this field. Hobbyists are a very intelligent group and are looking for the company and the product that offers more than the competition: consequently the entrepreneur with an idea who is willing to take a risk to bring his innovative product to the market is, in the long run, going to be a winner.

What were the most significant elements of the Faire? Why was it different from Trenton a year ago or from Atlantic City last summer? Well, of course, it was bigger, with more exhibitors. The sheer size of the hobby, the attraction to the general public who were not directly involved in the hobby but were interested enough to come to the Faire and find out what was going on. The tremendous diversity of the sessions which dealt with people and computers, legal aspects of personal computing, art and music, video art, computer systems for small businesses, computer networking, speech recognition and synthesis, amateur radio in computing, and multi-tasks on home computers. Also an extraordinary variety in the home-brewed exhibits. People who have taken electrical components, bits and pieces from the commercial kits and assembled them into not just a breadboard computer but a living, working, breating home computer for some purpose, whether it be for video art or some other purpose.

As Jim Warren said earlier, we can expect a repeat next year, only bigger and better. I would expect other Faires or conventions or festivals also to be bigger and better as the hobby grows and flourishes. So watch these pages for notice of the next one and go yourself! ■

★★★★★★★★★★★

Was a telephone computer so bold
The first of it's kind that was sold.
When you call its number
It is a bummer
It always puts you on hold.

There was a computer from Clyde.
Was indeed so very dignified.
We did not agree
On its pedigree.
It told me to kiss its back side.

There was a computer from South Bend.
That figured income tax without end.
Calculated the tax
A little bit lax.
Who is it they will apprehend?

Featuring on-the-spot interviews with two manufacturers, two retailers, and a publisher

Gamboling in Atlantic City

David H. Ahl

Atlantic City: Gambling has been approved, although it will be another nine to twelve months before it gets under way. Nevertheless, the beaches were a little bit cleaner than last year, the boardwalk traffic a bit brisker, and lots of construction was underway. On the other hand, the Shelburne Hotel, site of the "Personnel Computing '77" was as seedy as ever, the air conditioning in the exhibit hall spasmodic, the elevators generally out-of-order, the help surly and unpleasant. So the convention flopped? Wrong! Through the extraordinary efforts of John Dilks, Dave Jones, their wives, and other organizers, PC '77 was a fantastic success!

Creative sponsored a wine and cheese reception for the 400 or so conference and exhibitor people on Friday night. We poured the last people out around 3:00 A.M. Nevertheless, most people appeared for duty in their respective booths at 9:00 A.M. on Saturday. And a good thing too. All day Saturday and Sunday, virtually every exhibit was five-deep with people. The workshops and presentations were all crowded to overflowing. Anyone who went and said they did not have the experience of their life was either lying or asleep.

Heathkit showed their lines of computers in public for the first time. Commodore had a bunch of Pets. MITS/Pertec had a huge, showy booth. SWTPC had disk systems. OSI showed their hard disc. TDL had Xitans, Cromemco had Z-2s with built-in disks, Midwest Scientific had their new computer, and on and on ... Even Teletype was there.

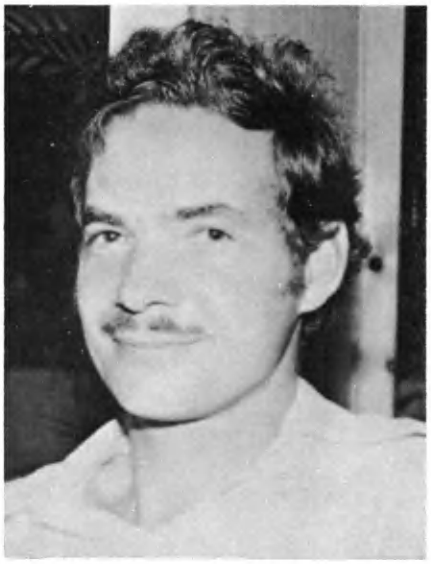
In the bedlam, I managed to talk to a cross-section of industry notables: a publisher, two manufacturers, and two retailers. Lightly edited transcripts of these conversations will be found on the next few pages.

Next year the exhibits will be in Convention Hall and some of the new hotels will be finished. So, put PC '78 on your calendar now ... the last weekend in August. It will be dynamite!

Adam Osborne

Adam Osborne is founder and president of Osborne and Associates, Inc., in Berkeley, California. He is a consultant on microcomputer systems, and author of the popular books, *An Introduction to Microprocessors, Vols. 1 & 2.*

Ahl: In talking to people here in Atlantic City, I've been trying to get some kind of consensus of where the hobby is going from here. Lud Braun felt that the days of the hobbyist are over and the days of personal computing are coming in with the advent of the Pet and the Radio Shack computers. In other words, home computing is kind of paralleling the early days of amateur or ham radio. From 1900 to 1920, there were only amateurs; then, all of a sudden, commercial radio appeared and brought with it explosive growth. The amateur hobbyists stayed around but had relatively modest growth compared to commercial radio. Do you think this will happen with personal computing? Where do you see things going?



Adam Osborne

Osborne: Personal computing, in which people have a computer for the sake of being able to program it, is something which will always be difficult to do, so it is never going to become a mainstream thing. It is never going to reach the levels which would interest Sears or J.C. Penney.

Ahl: Do you feel that applies with personal computers such as the Pet or Radio Shack machines?

Osborne: Anything that a guy has to program himself is always going to have a relatively limited audience. Now, when I say relatively limited, it can still amount to two or three hundred thousand people across the country, but this is generally insignificant. There are going to be computers going into the home in a form that you do not have to code. So, if whatever the computer is going to do for you is already programmed, that could run into the millions. You say, well, that is not the same thing, but I argue that eventually that is the way all computers are going. Most mainframe machines will reach the point where you won't have to program them; You will just have to define what it is you want them to do, thus ultimately reaching a merger of the home-appliance-type computer, which you don't have to program, with the mainframes which you also don't have to program.

Ahl: Does that imply different kinds of languages, or will everything be almost a black box from the user standpoint?

Osborne: Eventually there will be no languages. You will be programming by example. You will create on the screen an image of what it is you want it to do, and the extent to which there is any programming language will be the manner in which you define the variables. In other words, there can be no ambiguities; you must be given some means of identifying and resolving ambiguities. That is the extent to which there will be programming languages. But that will probably be 15 to 20 years away.

Ahl: That sounds like a long while. What is coming sooner?

Osborne: In the meantime you are going to see the more traditional programming languages around. I do not think you will see new ones very much. You will see them come, but I don't think you will see them catch on, because what will eventually wipe them all out will be this concept of an easily-programmable computer.

Ahl: Well, there are things like the high-level language that goes with, say, the Software Technology Music System. It is very easy to learn. It seems to me that there would be a lot of things like that, small languages for special-purpose applications.

Osborne: Yes, but that is almost beginning to be the programming example that I described. You write music in a form on the screen, and that creates for you the programs which generate the music. That is the beginning of the approach. It is not really a programming language any more than writing English is a language. Writing music notes is a language and what they are doing is writing music notes and then letting the machine recognize and play them.

Ahl: What kind of significant hardware do you see coming out in the next five years or so?

Osborne: I think you are going to see very inexpensive and very complex electronics. You are going to see the LCD display, the liquid crystal display replacing the touch switches, replacing the keyboards. At the end of the five-year period we are going to see great pressure put on floppy disks and rigid disks by memory devices ... such as CCD, bubble memories and other non-volatile types. Perhaps a little later, we are going to start seeing some real impact on printer technology from laser technology. It's expensive today, but in five years it's going to start becoming quite inexpensive. The laser creates a static charge so that a dry ink can be deposited on it.

Ahl: Like xerography?

Osborne: A xerographic-type approach. Xerox is doing that right now but the price is high and the quality relatively low which is the way all these things start. Then the quality becomes high and the price becomes low.

Ahl: Where do you see your role, Osborne and Associates, in the future?

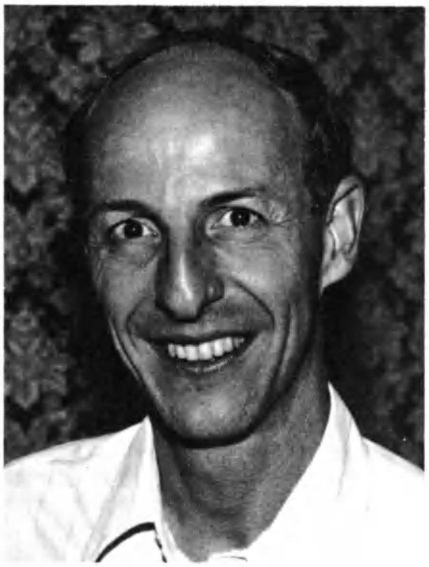
Osborne: I think there is going to be an ever growing need for a link between the technological side of society and John Q. Public who has got to live with that technology, and that is the need we are going to continue to address. We are going to be the translators; we are going to allow the public to understand what they must understand in order to live successfully in a technological society. For example, right now, computers have run wild in our society for the simple reason that no legislators understand computers well enough to draw up sensible laws, and the public at large will accept anything they are told because they don't know any better. I want to curb that. In the future we are going to carry on in that direction, to inform the public in terms that they can understand as to what is going on in the technologies which are affecting their daily lives.

Ahl: Do you see yourself staying mostly in the publishing business or branching into other means of communicating?

Osborne: I see myself basically staying in the publishing business — because right now that is the effective low-cost medium. We have not yet reached the point where video cartridges or anything else is nearly as good as visual graphics. It is going to be some time before any new technology comes along to replace the book.

Ahl: Adam, thank you very much for your thoughts and ideas.

Robert Suding



Robert Suding

Robert Suding is founder and president of the Digital Group in Denver, Colo. Digital Group Products have had an image of being "different"—non-standard bus, large circuit cards, naked hardware (in the beginning), etc. But one other thing stood out—high reliability, hence assuring the Digital Group a place in this rapidly expanding market.

Ahl: In your own perspective, where do you see the home-computer field going in the next couple of years?

Suding: Well, I see them probably trying to define more definite applications. I see a trend away from games, other than just for beginners

COMPUTER MART OF NEW JERSEY

The exhibits were mobbed from Saturday at 9 AM to Sunday at 5 PM.

and looking to more serious applications. One field that I am trying to get a number of people directed into is using microprocessors for handicapped people. So I am starting a new organization which will be amateur-computer fans designing and building and implementing projects for handicapped people. We have also set up sessions for various conferences next year showing how the amateur computer fans have implemented things for handicapped people.

Ahl: What specific things are you working on?

Suding: We are working on voice synthesizers for blind people that will allow a blind programmer to have a self-sufficient job using a voice terminal instead of a video terminal. We are working on devices to allow people with cerebral palsy to communicate through specially programmed input or output devices which could speak to them or it could type for them. It is something that is not of any one manufacturer, but something that almost any one of the various hobbyist microprocessors are far more than capable of doing.

RCA showed their new COSMAC VIP microcomputer.

Ahl: I heard you mention before that the Digital Group is planning to offer a speech synthesizer. How will it compare to some of the others that are available today?

Suding: The main thing about our speech synthesizer is that it uses very low programming overhead; it takes roughly 1200 cycles for every 100 words spoken. The reason for this is because it has a relatively limited total flexibility; it has no inflection and it cannot sing. It does not do anything but produce 64 different human sounds. The job of the microprocessor is merely to link the 64 different sounds into the desired language whether it be Russian or Spanish or English or Latin or any other language.

Ahl: Is it likely to come out with that strong Swedish or Slavic accent so characteristic of the earlier Votrax units?

Suding: Yes, this will definitely have more of a Swedish accent than most here. I heard some interesting rumors on how that originated; I suspect they are not true any more than any other rumor in hobbyist circles. I understand that it was invented by a Swede in Wisconsin, who developed the original algorithms for speech synthesis in the way he thought English was spoken.

Ahl: What other kinds of things, other than in the area of the handicapped, are you working on, or do you see being in the future?

Suding: Well, I am personally interested in energy management in homes and I am personally designing a system for my solar-energy house in Colorado where we actually control the whole house under process control. Existing solar systems are typically implemented using just standard relays and hard logic and temperature-differential thermostats. The microprocessor introduces a rather new element, and that's the possibility of programming the thing in the language of the homeowner, making it possible to use a "home management" language as opposed to Fortran or assembler and so on. The person talks to the thing and interacts with it by the machine asking how many watts he has in the system that he wants to control, what temperature he would like in the various rooms. He will be able to custom-program the house if he has visitors, or if there is a party; he can set it up to take into account various differences at his convenience. It offers some tremendous possibilities for energy conservation because it greatly enhances intelligence and record-keeping. I will be able to realistically evaluate different overall systems for using the minimum amount of energy in my house. I project that the cost of this processor will be quite a bit higher in my case, since I will have a lot more power, but a processor for a "normal" home controller being built now will be

SFS WALLETSIZE

2005AD of Philadelphia surrounded a three-screen video display running LUNAR LANDER with all sorts of real cockpit controls and had continuous lines of kids waiting to get at those controls.

about $300 or $400 quite easily.

Ahl: What about all the sensors required? Won't they be a lot more costly than the processor?

Suding: Probably not, because there are several new sensors recently announced in the one-dollar range. You also have some older temperature thermisters or transducers in the range of about $3 to $4 surplus that are very excellent. By using thermisters and analog multiplexors and something as simple as additional voltmeter chips and some old surplus telephone cable, we could probably wire up a complete house with the temperature input system and multiplexors in the range of about $100 for all the equipment; it all depends on how sophisticated you want to get. And then for control you can have spot room control and use little butterfly dampers that are run by a geared-down motor which you can buy surplus for about $3 or so and modify standard ductwork to have all this programmable. You should be able to justify the cost of the whole system in a year or two in what you save in heating bills.

Ahl: It sounds like you have to be a pretty good mechanical, electrical and civil engineer to implement such a system today by yourself.

Suding: It certainly helps but you would be surprised at what can be done. The biggest problem is finding people who know something about heating and solar and thermodynamics and who also know programming and computers and systems design; you can imagine these people are somewhat hard to find. That is why I say probably the best solution would be to develop a training language in terms understandable to a heating construction contractor.

Ahl: Do you find today that most of the customers of the Digital Group are relatively more knowledgeable hobbyists than in the past, or are they businessmen or what?

Suding: Much more knowledgeable hobbyists. The typical hobbyist no longer is one that will buy anthing. He is very, very knowledgeable. He has friends who have recommended things; I believe most systems are sold through a person who has a friend who has a system like it. We continually found, especially in the earlier days, we would send a system out to a given geographic area and within a couple of weeks we would get a couple more orders from that same area. It was obvious that the person building the system, especially if he got it running immediately, would call over a friend to watch it and he would be so impressed he would send in an order. It's a major investment and people just don't like to make major investments without a fair knowledge of the company and the product. We find that our biggest problem now is not being able to meet the demand; I suppose I can rationalize that it's a stabilizing influence in that it smooths out production.

Ahl: Well, there certainly are worse problems!

Daniel Meyers

Daniel Meyers

Daniel Meyers is founder and president of Southwest Technical Products Corporation in San Antonio, Texas, SWTPC originally made high-quality low-cost hi-fi amplifiers and other electronic kits. In the fall of 1975 they introduced the SWTPC 6800 computer kit in the same tradition and it has seen a wild, explosive growth ever since.

Ahl: I see you have some disk-based systems here and also that your components have nice pretty covers on them. That is a change for SWTPC!

Meyers: Well, one of the features that people had been asking for on the old box was a cover. So we were nice to them this time. We went ahead and put the cover on. We are doing what everyone else is doing, of course. We are expanding our systems, we have disks now, we have terminals, we have a printer and lots of new little inner goodies to go in the box, we have an interrupt timer now, we have a calculator interface, we will have an A-to-D, and also a PROM programming card that goes in the interface slot.

Ahl: The A-to-D, how many channels will that have?

Meyers: I don't remember, 6 or 8. It's designed around the National chip.

Ahl: Will you put some hooks in Basic to support these peripherals?

Meyers: Oh, yes.

Ahl: I notice a lot of companies have come out with grand schemes and pieces of hardware and then no way of using them, even with their own software.

Meyers: No, no, we definitely know about that need and we are going to take care of it. We will try to make the things useful to people. That was the philosophy behind, for instance, the little interrupt timer we recently introduced. A lot of people have oscillators that you can hang in their box, but you have to slip a switch to get different timings. OK, with ours you

Commodore showed five real, live PETs.

The Shelburne
The Shelburne

John Mauchley leaves banquet attendees (including John Dilks) with some words of history and wisdom.

don't do that. Ours has a software programmer at the divider module and you can get anything from a microsecond on up to an hour, software-controllable.

Ahl: Can you use it in high-level software, such as BASIC?

Meyers: Yes, you have to write it in using the instructions that go with that timer, in a user-defined machine language subroutine, but that's provided for in our BASIC. All of that is described in BASIC and the interface, of course, has complete instructions on what you want to tell that thing. If you wanted to change your timing intervals in the middle of the program, you could. It's very useful!

What else are we doing? Well, of course, we are trying to stay one step ahead of the Radio Shack wolf.

Ahl: Do you see Radio Shack and Commodore as being threats, or enlarging the industry and making more of a market?

Meyers: Both. Radio Shack could be a definite threat but Commodore, I don't really think so.

Ahl: Why is that? Why do you differentiate the two?

Meyers: Well, if you look at them, the Commodore apparently is very difficult to add on to and to interface with external devices; it costs more to expand the memory than our things do, or anyone else's for that matter. $300 or something ridiculous like that, is what I heard to add 4K. And the keyboard is unusable for touchtyping. Anyone who bought it should be happy with it for at least a week, before they wanted something better. Radio Shack, unfortunately for companies like us, has a pretty good product. It's got a creditable keyboard, it can be expanded quite easily, and it could definitely be a very competitive system.

Ahl: Although by using a non-standard bus, it may limit the number of peripherals.

Meyers: However, if they have the peripheral available, the purchaser will buy it. And that's just the point: they apparently intend to get the peripherals, so we'll see on that one, I don't know. Commodore, I feel, will get some people interested in computers, but they won't be much of a competitive threat as far as what we're selling.

Ahl: How do you see Heathkit?

Meyers: Heathkit has said they're not going to sell theirs assembled; they sell them by mail and through their stores, of which there are a limited number. We intend, by the end of the year, to be offering assembled units. I don't think that anybody can sell kits for business applications; it's just not meeting the need.

Ahl: You said business applications. Do you see more of your machines going into that kind of use?

Meyers: Well, I hope so. We're working towards that. We feel right now that we have the disks; we manufacture really everything that you need for a reasonable business disc system other than a printer and we're looking for one of those, but we haven't found the right mechanism yet. But that's the only item we can't supply currently. We are also working with a guy in California who has developed an excellent, very fast BASIC compiler. BASIC is lovely, but BASIC interpreters are ridiculously slow for business. Anyway, we have the disks and we have the language at an adequate speed, we have a more-than-adequate computer, so I think we're close.

Ahl: What about support in the business market? Don't they need more hand-holding and support? Do you expect your dealer network to be able to provide that?

Meyers: Some of them can, some can't. The fact that the language is BASIC is a big help in my opinion. There are a lot of application programs available now in BASIC. The first people, I think, that are really going to use these things in

Dazzle doodle (electronic Etch-a-Sketch) was a continuous attraction at Creative's booth.

business are the ones that have been here looking; the lawyers, the doctors, the accountants, the ones that understand electronics and are interested. Later you get to the corner ice-house and those places where you have to walk in and set it on the counter and say, "This is a diskette and it goes here, and you push B and the system starts," and so forth. It's going to be slow. I don't agree with any of these people who say we're right on the verge of this huge market, and it's going to grow into this colossal industry in a year or two....

Ahl: Some people have said that this is the year that will separate the hobbyist market from the home and small-business market, the hobbyist is essentially finished, and future expansion will come from business systems.

Meyers: There's only one thing wrong with that. My opinion is that *nobody* at this show has got a system with the possible exception of MITS/Pertec that could be used in a small business. Period. I just don't believe I've seen anything else here that would be suitable, because they're lacking either the language, or reliability, or something. Any one of them you want to name has got a problem somewhere that is going to keep them from being successful in selling their system to businesses. And they don't really understand that. I think *I* understand that because *I've* used a computer in *my* business for 3½ years now and I know what my problems are, I know what I've got to have, and I don't have a system that I could use in my plant yet.

Ahl: I have to tell you this. We have five computers at *Creative Computing* and I don't use a single one of them for the business.

Meyers: Right. We're just not quite there yet. But we're on the verge of being able to sell a system that would work very nicely in a smaller business. I don't think anybody here today, with that one possible exception, could say that they have an adequate system for small business, so we haven't even started getting into that. It's going to be a long, rough road. If that's where people intend to make their money, I think they're going to be in for a big shock when they try.

creative computing
THE NO. I SOFTWARE AND APPLICATIONS MAGAZINE
SIGN UP FOR YOUR FREE ISSUE
WITH EACH NEW SUBSCRIPTION OR RENEWAL ORDER

Some of the Creative Computing barbeque set, l to r: Alan Salisbury, Burchie Green, Linda Eckerstrom, Steve Gray, Sandy Ahl.

Alan Hald

Alan Hald is founder and president of Byte Shops of Arizona and Byte Shops Mail Order. The Byte Shops franchise has had a rocky history, but Alan's store and mail-order operation with its outsize, popular catalog has clearly been a success right from the start.

Ahl: The first question is: can the Byte Shop of Arizona find happiness in New Jersey? Seriously, why did you decide to exhibit here?

Hald: We have Byte Shops Mail Order which is nationwide, and we have a lot of customers on the East Coast. The Byte Shops of Arizona run the Byte Shop Mail Order. We also handle Micro Age which distributes to a lot of independent small stores and a lot of those customers are also here on the East Coast.

Ahl: Where do you see things going from here in the next year or two?

Hald: It's hard to see our two years, but what we will see next year is the beginning of full-fledged systems stores. We are going to be involved in systems for various businesses and occupations, and a continuation of the growth of the hobby stores, and the beginning of the mass retailing of low-cost systems. However, I think it's difficult for a lot of the stores to get involved in the mass market mainly because of the demand and the way the companies like Commodore deal with the products. They are looking for the very-large-volume buyers.

Ahl: Do you see Commodore and those products being a threat to the independent stores?

Hald: No, I think what will probably happen is they'll open up the market even wider than it is now. It is difficult for us to appeal to a wide range of the public when you are talking about systems that cost $1,000, particularly assembled systems. There is a tendency now for people to want assembled systems rather than kits. Once that price barrier is broken, there will be more people coming into the market. They will take one home and use it, and once they start learning about computers, they will want one with more capability. So people will become more intrigued with adding on to their systems or exchanging systems and upgrading them. They will probably be disappointed to find they cannot add anything on the Pet like a disk drive and they will come in and look at a North Star and see how convenient it is to use, and they will start looking at S-100 types of systems. But I think if someone like Commodore is very successful, maybe other larger corporations will say, "If they can do it, we can," and that, in effect, will open up the distribution channels. If that is very successful—

Alan Hald

let's say Commodore hooks up with a large distributor like Sears—then someone like Sharp will come in and wonder who else to distribute to, and I would say the computer stores are a natural since they are the hobbyist place and they will be hungry for products.

Ahl: Well, I have noticed that a number of the computer stores today are carrying those little Microchess sets, as well as the Fairchild games and other prepackaged products. I see a kind of a cross-fertilization there.

Hald: Very true, what is happening now with product lines becoming so broad is that a lot of stores are going to have to start making decisions as to what type of store they are going to be. The capital requirements to support a store are limited. Some stores are moving towards systems and they don't want to be bothered with hobbyists; what they are trying to sell are the big systems. On the other side, some will develop into more consumer-oriented stores that handle shelf games, hobbyist type of computers, and the kits. I think we will see two types of stores emerging, and probably eventually a third, that being a mass-consumer type of store. It will be interesting to watch.

Ahl: It sure will!

Ken Greene

Ken Greene left a steady, well-paying job in industry in early 1977 to open a computer store in Morristown, New Jersey, the first of the ComputerLand franchises. He works twice as many hours a day now, but wouldn't give it up for the world.

Ken Greene

Ahl: What do you think of the show, Ken? Have you gotten business, or references? How is it of value to a retail store?

Greene: Well, we make contacts with people in our area who did not know we existed, and vice-versa. I believe the show is a really good place.

Ahl: What kind of people have you been talking to here compared to the kind of people who come into the store?

Greene: A lot more business-oriented people here than come into the store. There you get many more hobbyists. Here, people are looking for packages; they want an inventory package system, or some other business system. Big systems too, like $10 to $12 thousand.

Ahl: You mean here! At the show?

Greene: Absolutely. Very, very few people came by who were hobbyists. They may have come too, but they didn't talk. They came though and they said, "Gee, that's nice" and then left.

Ahl: That's interesting, because you would think that a show like this which was billed mostly as a hobbyist show, would not draw the business-type customers.

Greene: Well, I think next year it won't be anywhere near a hobbyist show. I think next year, we will see even the big business systems. I think we will see DEC in here with systems and all the others. Just from what the people are asking for. When we come down next year, we'll have six booths reserved. We may go with even more. Two of those booths will be my separate corporation which will display its business, hardware, software packages; the rest will be my store and other ComputerLand stores.

Ahl: In your own opinion, where do you think things are going in the future?

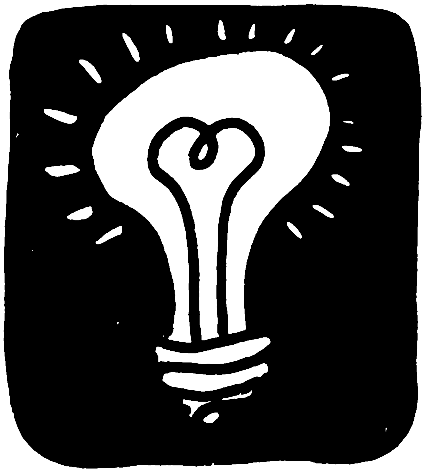
Greene: Everything looks like its going commercial—$500 Fortrans that require disks. Everybody wants a Fortran but the hobbyist isn't going to buy that. I think the beginning hobbyist is going to get sucked in by the Commodore Pet and the Radio Shack and the Heathkit. And what does that do to a computer store like mine? Well, those companies are just going to educate the market place. The only ones I consider high-line quality units are the Heathkits. And Heath will always command the hobbyist in the build-your-kit market. The computer stores like myself and the Computer Marts and so on are going to be more like a high-level stereo dealer. The guy who really wants to buy the big component system is going to come to us, rather than go to the department store or his local Radio Shack and buy their assembled units. So the future for us is mostly with small businesses and top-line hobbyist systems.■

INSPIRATION

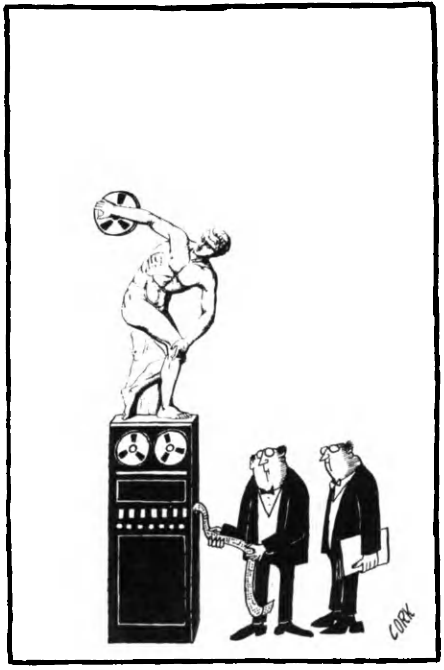
Electrons have minds of
Their own.
There is no lobotomy
For them.
There is no shock treatment
Short of a Cyclotron, and
Even that is not permanent.

In a bundle of electrons,
Some are tied to atoms
And some are not,
They are merely
Running around the brain,
Electro-Chemical impulses
Which now and then
Pause long enough to become
Poems.

Esther Gloe



AP-69



©Creative Computing

Some Tips On Using A TV Set For Computer Output

David H. Ahl

So you've bought yourself a spanking new SOL-20 that expects to provide output to a video monitor. Or you have a VDM or TV Dazzler of Apple-II or Poly 88, etc., etc., all of which put out a beautiful video signal for a video monitor. But unfortunately you don't have a video monitor.

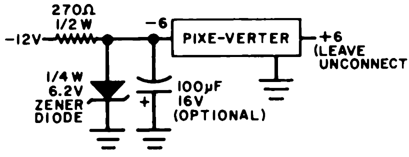
In this situation you have three choices:

1. *Modify a standard B&W TV* to act as a video monitor. Generally this can be done fairly easily by breaking the existing signal between the video detector and video amplifier stages and then inserting the new signal into the input of the video-amplifier stage. You may have to add one or more capacitors, diodes, and/or transistors to preserve the white and sync levels. If you want to still be able to use the set as a tv, you'll want a SPDT switch at the break in the circuit. It is *highly* advisable to use a transformer-type tv set with the chassis isolated from the power line. If you follow this route, you'll find the appropriate circuits in DonLancaster's book, *TV Typewriter Cookbook* published by Howard W. Sams, or in Don's article "Television Interface" in the October 1975 issue of *Byte*.

If you don't want to brew up the circuit yourself, Pickles & Trout makes a kit (TVM-04) for modifying a Hitachi TV set with the SX chassis (Models P-03, P-04, P-05, P-08, P-53, P-63, etc.) into a quality video monitor. The TVM-04 can accept a 0.6V P-P video signal as from a Polymorphics VTI or 1.4 V P-P as from a Processor Technology VDM. The kit contains a switch permitting use as either a monitor or TV set. The TVM-04 is available for $20 postpaid from Pickles & Trout, P.O. Box 2276, Goleta, CA 93018. In the New York area, a 12-inch Hitachi P-05 TV set sells for about $80, thus the total cost is $100 or so.

2. *Purchase a TV Monitor.* A good-quality B&W 9-inch Hitachi monitor (VM-909) sells for about $185. Larger-screen Comrac, RCA, or Sylvania industrial-quality monitors are in the $500 price range. Super quality if you can justify the expense.

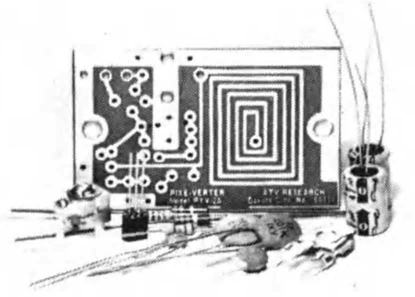
3. *Build an RF Modulator.* This device accepts a video signal and adds an RF carrier so that the signal can simply be fed into the antenna terminals of any standard TV set. For color output (from the TV Dazzler) this seems to be the best approach, since "monitoring" a color set is risky and color monitors are expensive. One RF modulator, the PXV-2A Pixe-Verter requires -6V (actually anything between -5 and -6.5 V) which can be furnished by batteries or from the computer. The device is so small (1.25" x 2.1" x 0.8") that it will fit inside the computer practically anywhere. Be sure it is inside the computer or in a shielded chassis box, otherwise your computer will be broadcasting to your whole neighborhood. Most computers have available either -12 or -16 volts of filtered DC. The following circuit will allow you to power a Pixe-Verter from this.



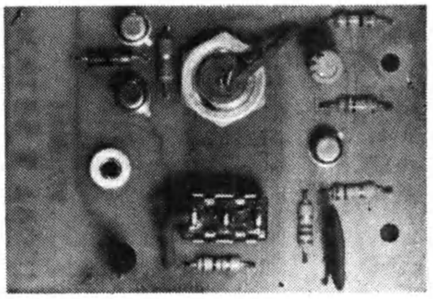
Change the dropping resistor to 470 ohms for -16 volts. In this set-up, the Pixe-Verter draws about 2 ma and the Zener diode, 20 ma. The Pixe-Verter is available for $8.50 postpaid from ATV Research, 13th and Broadway, Dakota City, NB 68731.

A similar device with some added goodies is available from M&R Enterprises. This one has an antenna/modulator selector switch, long cables, and operates from a wide voltage range (6-12 VDC). This RF Modulator is available for $25.95 postpaid from M&R Enterprises, P.O. Box 61011, Sunnyvale, CA 94088.

Another source of interest for those hard-to-find video cables, connectors, adaptors, etc. (as well as audio plugs, cables, etc.) is WIDL Video, 5325 North Lincoln Ave., Chicago, IL 60625. phone (312) 271-4629. ■


Pixe-Verter before assembly

M&R RF modulator kit


Pickles & Trout monitor kit

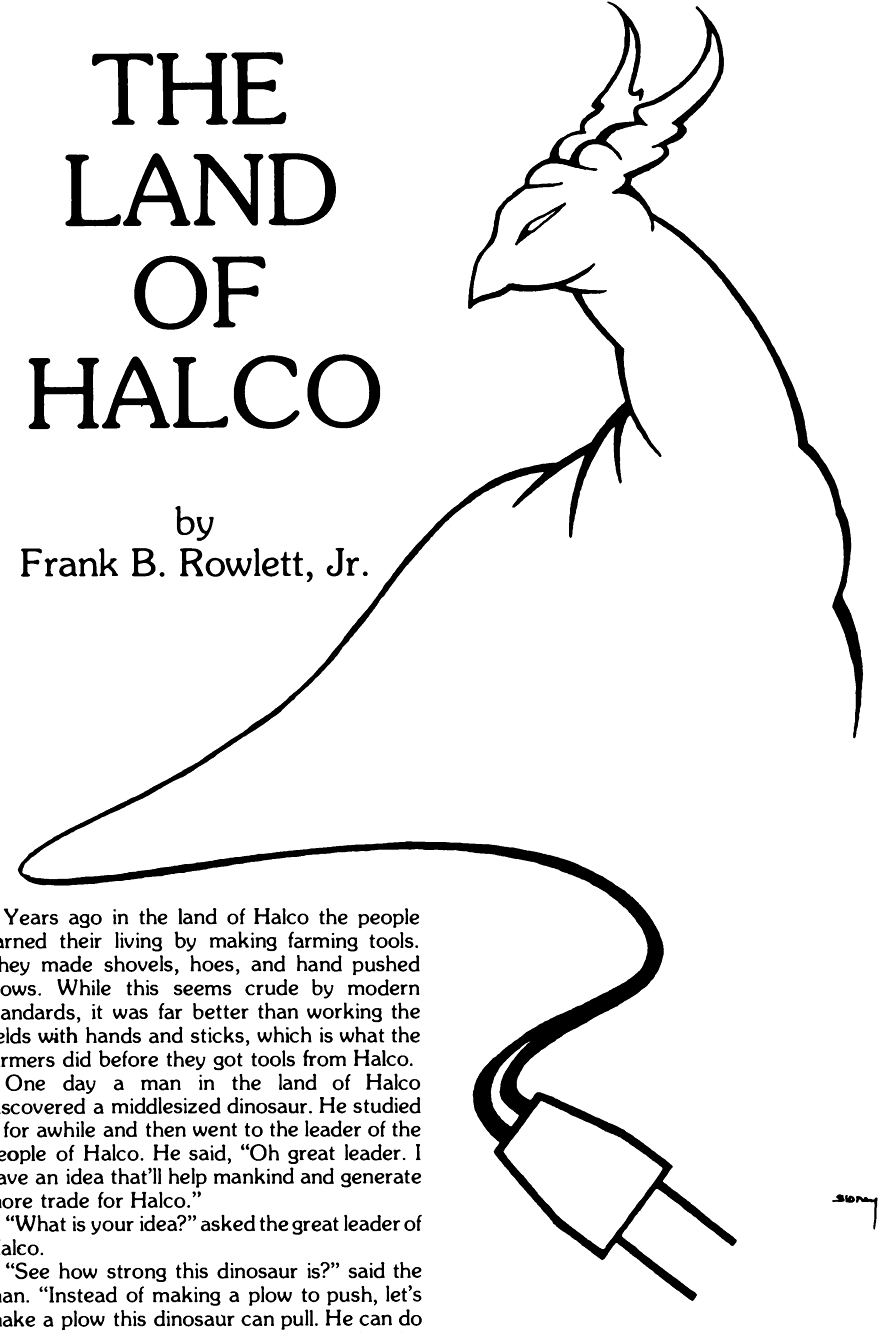
Fiction
and
Foolishness

Satire

THE LAND OF HALCO

by

Frank B. Rowlett, Jr.



Years ago in the land of Halco the people earned their living by making farming tools. They made shovels, hoes, and hand pushed plows. While this seems crude by modern standards, it was far better than working the fields with hands and sticks, which is what the farmers did before they got tools from Halco.

One day a man in the land of Halco discovered a middlesized dinosaur. He studied it for awhile and then went to the leader of the people of Halco. He said, "Oh great leader. I have an idea that'll help mankind and generate more trade for Halco."

"What is your idea?" asked the great leader of Halco.

"See how strong this dinosaur is?" said the man. "Instead of making a plow to push, let's make a plow this dinosaur can pull. He can do the hard work of plowing the fields for the farmer, and the farmer can then do more work."

"What good will that do Halco?" asked the leader.

"If it works — and I'm sure it'll work — then we can gather up all the dinosaur eggs and raise dinosaurs and trade them and the plows to other people in other lands," said the man. "Then we'll all have food and goods in abundance."

"That might be a good idea you have," said the leader of Halco. "I'll provide you some land, some plow makers, and whatever else you need to try out your idea."

"Thank you great leader," said the man, and he went immediately to work developing and perfecting his idea.

Soon the idea was perfected. The people of Halco began trading the small dinosaurs and the plows to the peoples of other lands. Halco prospered. The people of Halco were happy and richer than ever before.

After this had gone on awhile another man of the land of Halco came to the leader of Halco. He said, "Oh great leader, I have an idea about how to improve our dinosaurs and plows so that the people who own them can get more work done. The peoples of other lands will then trade more with us to get our improved plows and dinosaurs."

"What is your idea?" asked the great leader of Halco.

"Let's breed bigger dinosaurs to pull a two bladed plow. That way, the farmer can get twice as much work done in the same amount of time."

"That sounds like a good idea," said the leader of Halco. "I will give you dinosaurs, plow makers, land, and whatever else you need to try your idea out."

"Thank you great leader," said the man, and he went to work developing and perfecting his idea.

When the idea was perfected and the people of Halco announced it to the other lands there was great rejoicing throughout the known world. The people of Halco now made it possible for the farmers to double their production.

Of course, all this did not come for free to the farmers. They had to trade some of their crops to Halco to obtain the bigger dinosaurs and plows, but most could afford it — especially those that had traded for the early model of dinosaurs and plows.

For the farmers who couldn't afford to trade outright for a dinosaur and plow, the leader of Halco invented leasing. The farmer would give Halco a portion of his crops each year in return for the dinosaur and plow. That way all farmers could afford dinosaurs and plows, and the people of Halco had a steady income.

The farmers then began to notice something. They noticed that while the first model of dinosaur ate a lot, the new and larger dinosaur ate many times as much. Thus the farmers had to plant more land to feed the larger dinosaurs. They also had to provide special buildings for the new and larger dinosaurs because they wouldn't fit in their barns as the earlier models had. But this was all right because the farmers were now able to produce much more from their farms for the same amount of work.

Many farmers also wanted the newer model dinosaurs because they didn't want to fall behind in technology from the other farmers in other lands. Besides, it was nice to show visitors to the farm the "newest and latest" model dinosaur and plow.

Things went well for the people of Halco. They prospered even more. The leader of Halco knew a good thing when he saw it, and invented research teams to come up with new and better dinosaurs and plows. To make sure that farmers in other lands were aware of the newest and latest dinosaurs and plows from Halco, the leader of Halco invented marketing. To be sure that the Halco people engaged in marketing knew what the farmers would buy and to be able to get it quickly for the farmers, the leader of Halco had offices set up next to the farmers' fields. These were known as "field offices."

Now that field offices were established, Halco began providing service on their dinosaurs and plows. The leader of Halco invented the word "fixer" to describe the people from Halco that serviced the dinosaurs and plows. If a plow broke or a dinosaur got sick in the field, a fixer from a Halco field office would go immediately and fix whatever problem there was. These people were called "field fixers."

Things were good for Halco. They continued to prosper as they built bigger and better plows and bred bigger and better dinosaurs. The leader of Halco wasn't even bothered by some of the people of Halco leaving and starting their own lands where they bred dinosaurs and made plows. After all, Halco had been there first with the best. At least, that's what all of Halco's customers said.

All farmers knew that Halco would take care of them if they traded for Halco's dinosaurs and plows. The farmers weren't sure of the people from the new lands that were trading dinosaurs and plows — many farmers felt they were fly-by-night lands. Thus, the farmers didn't trade much with the new lands. That's the way Halco kept eighty percent of the dinosaur and plow market.

Whole new businesses sprang up based on the dinosaurs and the plows. There were special plow and dinosaur modification businesses (although the leader of Halco warned the farmers that did business with these people that Halco wouldn't fix their dinosaur or plow if it broke afterwards), there were people that specialized in getting a particular dinosaur or plow to do a special job in a certain field, and even the farmers began hiring people to specially handle the dinosaurs and plows.

Soon the specialists in dinosaurs and plows began telling the farmers they worked for (either directly or under contract to) that they needed bigger dinosaurs and plows. The specialists also said more power could be obtained by running two dinosaurs linked together. Besides, there was less chance of the plowing being interrupted then — there would be an immediate backup available until Halco could get there and fix the problem. Besides, if the farmer had a big enough dinosaur or several dinosaurs, then Halco would put one or more Halco people "on-field." Then the specialist would have someone to talk to in dinosaur and plow jargon.

Things kept going along and the dinosaurs and plows kept getting bigger and bigger. The farmers had to produce more and more to keep the dinosaurs and the specialists fed. Soon, many of the farmers found that the dinosaurs and plows were costing them too much for what they were getting in return.

When the farmers told this to the specialists and the people from Halco, the specialists and the people from Halco said, "Not true! We will keep tuning the dinosaurs and the plows and you'll get better performance from them. After all, Halco is constantly coming out with engineering changes and modifications, and we are making special on-field modifications for the particular fields you have to plow. Wait — things will get better. In fact, they'll get better sooner if you get the very latest model of dinosaur and plow that Halco has just announced."

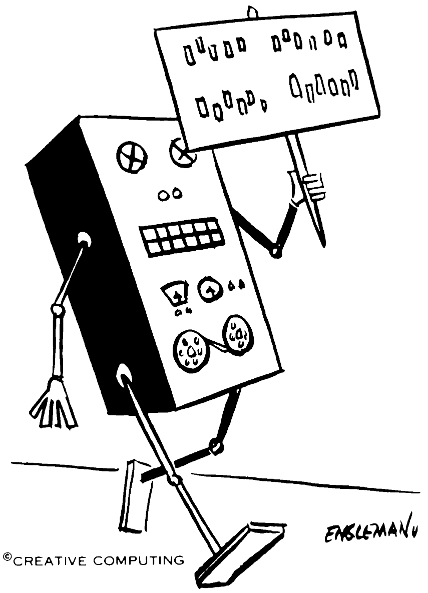
Some farmers were getting suspicious — they wondered if the specialists cared more for the dinosaurs and plows than they cared for the farmer and his crops. Halco didn't wonder about this — they knew. Halco kept promising and delivering bigger and better dinosaurs and plows to the specialists. Halco knew who now controlled trading for the dinosaurs and plows.

The farmers kept getting the specialists the newest models of dinosaurs and plows and watched and waited. Things didn't get better, but what could the farmers do? They couldn't stop now — what would happen to their farms if they didn't have the dinosaurs and plows to farm them with? Chaos was in the lands of the farmers.

Then one day a man came from a distant land with something entirely new — he called it a mule. The man told the farmers the mule could pull a plow better than a dinosaur, and it was cheaper to own. It didn't eat nearly as much as even the first model of dinosaur, and it took up only a small part of a farmer's barn. It didn't need a lot of specialists to handle it. It would pull a simple plow — something like the original plow the first model of dinosaur pulled, but without all the complicated hardware necessary for attaching the dinosaur to the plow. The mule only needed a simple harness. On top of it all, the mule could plow faster than most of the dinosaurs. It was exactly what the farmers wanted and needed. The farmers started buying mules from the man.

The man sold many mules and many plows. Soon farmers were throwing the people from Halco and their dinosaurs and plows and specialists from their fields — their farms — and their lands. The greatness of the land of Halco was over. The farmers worked their farms with their mules and made bigger profits. Everyone was happy except the people from the land of Halco and the specialists in dinosaurs and plows.

What happened to the dinosaurs? Why they died out — no one could afford or wanted to feed them. That's why dinosaurs are now extinct. At least most of them. ■



THEM HOBBYISTS

by
JIM DUNION

HOW 'BOUT THEM HOBBYISTS,
AIN'T THEY THE THING?
BUILDING THEM COMPUTERS,
OUTA' LITTLE BITS OF STRING,

LITTLE BITS OF HARDWARE
AND A WHOLE LOTTA LUCK,
TIME IT'S ALL TOGETHER,
IT COSTS A PRETTY BUCK!

THEM DAREDEVIL HOBBYISTS
LISTEN TO 'EM MOAN
TRYING TO BUILD COMPUTERS
RIGHT THERE IN THEM HOMES

BUYING THEM POLY-PAKS
BUILDING THEM KITS
SOME BUY A SPHERE
AND SOME BUY A MITS.

LOOK AT THEM HOBBYISTS
STRAINING THEIR WITS
PEERING IN THE INNARDS
LOOKING FOR THEM BITS.

LOOKING FOR THE LOOSE WIRES
LOOKING FOR THE CRACKS
COUPLA HOURS LATER
THEY'RE LOOKING FOR AN AXE

HOW TO BE A HOBBYIST
I'M GONNA LET YOU KNOW
GIT YERSELF A MICRO
AND WATCH IT START TO GROW.

COMPUTER CONTROL

BY MICHAEL R. VITALE

What happens when a computer designed to accept thought patterns as input tries to output thought patterns of its own?

```
999 END
READY
RUN
*** PLEASE STAND BY ***
```

The cursor blinked white against the green screen of the cathode ray tube. "Oh, no," the student thought to herself, "what now?" She glanced up over the terminal towards the glass wall of the computer room, attempting to see within any reason for the halt in service. She scratched the place behind her left ear where the electrode rubbed against her skin and adjusted the metal cap slightly. Her eyes returned to the cursor, flicked away, then returned and remained fixed. A few seconds later she rose, took the cap from atop her head, and set it down on the chair. She ran around the terminal once, flopped to the floor, and did ten pushups. Then she ran around the terminal twice, flopped to the floor, and did ten more pushups. Then she ran around the terminal three times, puffing slightly now, flopped again to the floor, and did another ten pushups.

After circling the terminal 55 times and doing a total of 100 pushups, the girl picked the cap off the chair, replaced it on her head, and slumped wearily behind the screen.

```
***CONTINUE***
```

appeared before the girl's glazed eyes. From somewhere behind the glass wall arose a deep chuckle.

* * * * *

The intercom buzzed twice. The president took his feet off the desk, set down the paperback book, and flicked the switch.

"Professor Eckert to see you, sir."

The president mentally ran through the names of the school's 253 faculty — *his* 253, as he liked to think of them — but could not place Eckert. "Professor Eckert?"

"The head of the computer center, sir."

"Oh, Eckert, What the hell does *he* want? If it's about the budget tell him I'm sorry but not another dime for his damn tape packs or whatever."

"He says it's an emergency, sir."

"I'll bet it's an emergency. You tell that twerp that if his damn machine has screwed up the payroll again I'll send him straight back to typewriter repair school."

"He seems quite excited, sir."

"He's probably finally figured out how to get that toy of his to do some useful work for a change. Give me two minutes." The president pushed aside several overflowing manila folders, some old issues of the campus newspaper, a half-eaten chocolate bar, and a report from a dean who had left the college seven months before. He picked the top binder from a large pile and opened to a page at random. He flicked the intercom switch again. "Send Eckert in."

The door opened and a small, nervous man with wildly tousled hair took a few steps into the cavernous office.

"Well, Eckert, good to see you! I've just been reading over your latest outprint."

"Printout, sir."

"Well, whatever. I see that the number of students has increased 2.84736 percent over the past three years, while the number of faculty has decreased by 7.5846 percent and the number of administrators has increased by 14.596 percent."

"That's true, sir. Actually I came to discuss something else."

"I hope it's not the budget again, Eckert. I *am* sorry about cutting you back this year, but I thought that with that large grant from the National Science Foundation you could get by without so much help from the college."

"It's the NSF project that I wanted to talk about, sir."

"You've made progress, I hope? It's almost time for our annual report to the government."

"Yes, we've made progress." Eckert sighed and gazed at the carpet.

"Well, what is it, then?"

"As you recall, sir, the grant was to support an experimental project in artificial intelligence."

"Of course I remember." He didn't. "I thought at the time that developing some artificial intelligence among the faculty was a good idea, since they seem to be somewhat lacking in the natural kind."

Eckert forced a smile. He had heard the joke before. "We were trying to work out a means by which instructions could be entered directly into a computer without the necessity of punching cards of typing on a terminal. It's a programmer's dream!" Eckert's glowing eyes moved briefly upward, but were forced back to the carpet by the president's uncomprehending stare. "During the early seventies, various means of entering information by voice were developed, but . . ." The phone rang.

"Excuse me a moment, Eckert." The president picked up the receiver. "Yes?" The president listened for a moment. "Well, you tell that bastard that he damn well *better* get those lawns cut, and by this afternoon, or I'll fire his ass." The president listened again. "Well, I suppose you're right. Talk to the union steward, then. Ask him to be reasonable, for once." The president slammed the phone back into the cradle. "Dammit, nobody ever does *anything* around this place. Now where were we?"

"I was explaining the background of the NSF project."

"Oh, right. Go ahead."

"The vocal techniques worked fairly well, but they were very sensitive to slight changes in tone — a strange voice threw them off completely, and even a sore throat could cause trouble. Besides, they were *slow*. A human can think much faster than he can speak."

"Seems to me we've got quite a few people around here of whom exactly the opposite is true."

Eckert forced another smile. He had heard that one before, too. "In any case, we wanted to develop a technique for entering information directly from the brain into the computer. No typing, not even any speaking. So we formed an artificial intelligence lab, applied for a government grant, and began working. That was three and a half years ago."

"Yes, you took over the old bus garage, didn't you?"

"The old boiler room, sir, when the college installed the solar panels. But we've gotten most of the soot out by now. You'd hardly recognize the place. Anyway, eventually we developed a sort of metal beanie through which the brain could transmit information directly to the computer. The data channel was designed to be one-way — the computer could send information back to a fast terminal device, as usual." Eckert paused, sighed, and resumed speaking. "We gathered some student volunteers, all experienced programmers, and turned them loose with the beanies. Until last week everything was going fine."

"Yes?" The president knew better then to ask directly what had gone wrong. He passed his hand over his forehead, trying to wipe away visions of dazzled students with hair standing on end and eyeballs rolling idly.

"Then we began noticing that the participants were behaving strangely. First they began volunteering to sweep out the computer room."

"Sounds good to me."

"Yes, but not every hour. Then they began offering to oil the tape drives, vacuum the disk packs, and empty the chad box on the card punch. Finally we had to lock the machine room door so we could use the equipment."

"What are you suggesting, Eckert?" The president's tone had changed.

"Well, apparently the computer has grown . . ."

"GROWN?"

"Well, developed, then, another circuit, which somehow lets it program the students." Eckert stopped.

The president tried to absorb the news. "So we load the thing with calculus, plug in all the students, and cut back the math department? Well done, Eckert!"

"No, sir, I'm afraid that's not exactly what I wanted to say."

"What's the problem, man?"

"You see, we don't know how to control *what* the students are being programmed to do. The computer seems to be doing it on its own."

"Son of a bitch." The president savored each syllable. Having the payroll screwed up was nothing.

"I wanted to alert you to the, uh, situation as soon as I was sure that it was really happening. I have my best programmer working on it right now," Eckert lied.

"Your best programmer? Why can't you do it yourself?"

"Well, I'd like to, but I have the daily, bi-weekly, weekly, semi-monthly, monthly, quarterly, and annual reports to get out, besides the new grading system and the classroom space evaluation."

"Of course. Well, all I can say is you'd better get some action . . ." There was a scuffling noise in the outer office, the sound of a body being pushed against a wall, then the door was jerked open. A large, bearded student ran across the room, pushed a custard pie into the president's face, and left as quickly as he had come.

* * * * *

Eckert pulled open the door of the computer center, pushed past a student doing frantic jumping jacks, and edged between two girls playing a game of imaginary tennis. He entered his secretary's office. "Has Tom arrived yet?"

"He's waiting for you inside."

Eckert tried to open his door. The knob came off in his hand. He sighed. "The door is still broken."

"I know. I've asked to have it fixed, but there's no . . ."

No money in the budget, Eckert thought, and rapped on the door. It was opened from the inside by an emaciated, acne-scarred student wearing pants several inches too short. He was rubbing sleep from his eyes with his free hand. "Hello, Tom. Thanks for coming over. I know you were up late last night straightening out that problem with the Fortran compiler."

"I found a bug at 04756 octal — a zop fault which under certain circumstances prevented the J-block from linking up with the permfile."

"Well, thanks for fixing it." Eckert carefully closed the door behind him. "Tom, I wanted to talk to you about the artificial intelligence project."

"You haven't let me work on that one yet."

"I know, Tom. I thought we could use you more effectively somewhere else." The true was, Eckert mused, that Tom seemed to relate a lot better to machines than to anything which involved people. "But we need your help now." Eckert recounted the problem in detail, emphasizing the machine-language programming techniques which had been used. "We've got a list of all the students who participated in the project. Of course we'll have to scrap the whole thing now, but first we have to find some way of, well, de-programming them. That's where you come in. Do you think you can handle it?"

BLAH BLAH
BLAH BLAH BLAH
BLAH BLAH

"I'll try." This was Tom's way of showing enthusiasm. "Can you give me access to the documentation?"

"Yes. My secretary has all the information on her desk."

"OK. I'll try to find the problem. Then we can round everybody up and unload them all at once. I'll be in touch." Tom reached for the door, ignored Eckert's warning cry, and pulled off the other doorknob. He slammed the door, locking Eckert into the office, took a large pile of jumbled papers off the secretary's desk, and left the center.

* * * * *

Eckert stood nervously behind the control console. "Are you sure this is going to work?" Tom did not bother to look up from the keyboard.

"We're ready to go."

Eckert left the machine room and surveyed the small group of waiting students. "All right, everyone, we're going to try a new type of experiment." Eckert felt a momentary twinge of guilt for not having told the students what had happened — or what he hoped was going to happen now. "Please put on your beanies and sit down behind a terminal." The students scuffled noisily into their places, and Eckert walked behind the row of terminals to be sure that everything was ready. He signalled to Tom through the glass. Tom typed a few characters and looked up expectantly.

* * * * *

There was a knock at the dormitory-room door. Tom looked up from the latest issue of *Computing Reviews*, glanced at his watch, and said, "Come in, Cindy." The door opened, and a lithe, tanned girl with long brown hair and a perplexed expression entered. Cindy removed her pullover, kicked off her shoes, and slid out of her bluejeans. Tom got up from the desk, locked the door. and switched off the light. "I've been expecting you." ■

HOW I INSTALLED A YELLOW COMPUTER AND SAVED 50¢ A WEEK

by Alex Ragen

Our firm, Amalgamated Paper Bag Company, is a small manufacturer employing two dozen production workers and four clerks, and is located in a rather remote part of the Brooklyn waterfront. I was quite surprised then, when my secretary (who is also my bookkeeper and my wife's second cousin) told me that an I*M salesman was waiting for me in my office when I arrived one morning.

"We're a very small company," I explained politely. "I don't think we can justify a computer here."

"Sir," he answered. "I suppose you haven't heard about this morning's historic product announcement — a new series of small business computers designed especially for firms like yours."

"But our firm is *very* small, I don't think —"

"Our biggest small computer is the System 1/3," he interrupted. "For smaller businesses we have the System 1/32, which fits into a desk drawer, or the System 1/360, which fits into a standard business envelope, or the System 1/370, which fits into a standard ball point pen refill."

"But —"

"As an introductory offer, valid for ten minutes only, you can have a System 1/32 with 16 bytes of memory, three tape drives, six disk drives, two high speed printers and all the programs you can compile in a day for $1 per week for the first year. Extra memory, controllers and software are not optional but do cost more. Sign here please, and don't forget — the offer expires in six minutes."

"But what about people to operate the computer?"

"Do you have a B.A.?"

"Yes, in anthropology. But I don't know a thing about computers."

"It doesn't matter. We'll enroll you in Columbia for a doctorate in computer science. In no more than three years, you'll know everything there is to know. We'll even get you a discount on the tuition, and in the meantime we'll supply the programmers and operators at a small additional charge. Anyway, you can always use our standard packaged systems which already work just fine for thousands of small businesses just like yours."

"But there is no business *just like* mine," I protested. "And how could these programs be working already if the computers were just announced today?"

"They were simulated on other computers, of course. I see you still have a lot to learn. The offer expires in thirty-four seconds, sir, so I'd appreciate your John Hancock right here."

What can I say? The lure of being the first of my competitors to plunge into the twentieth century was too much for even a hardened businessman like me to resist. I signed everything, and a month later the computer arrived.

"Now you can fire your bookkeeper," the salesman said as the computer was being unpacked. "The computer will do all of her work and you'll save a bundle."

"Are the programs ready?" I asked.

"Not yet," came the confident reply. "But they will be tomorrow. Now let's get to work on the programming specs."

But the programs weren't ready on time. To the salesman's credit, it wasn't his fault. As he later explained, the computer was so new and revolutionary that the systems software (I didn't know what that was then but I do now) wasn't entirely reliable. It seems all that clever simulation on other computers hadn't been so clever after all. But, as the salesman said, this was something that nobody could possibly have foreseen at the time. So nothing worked.

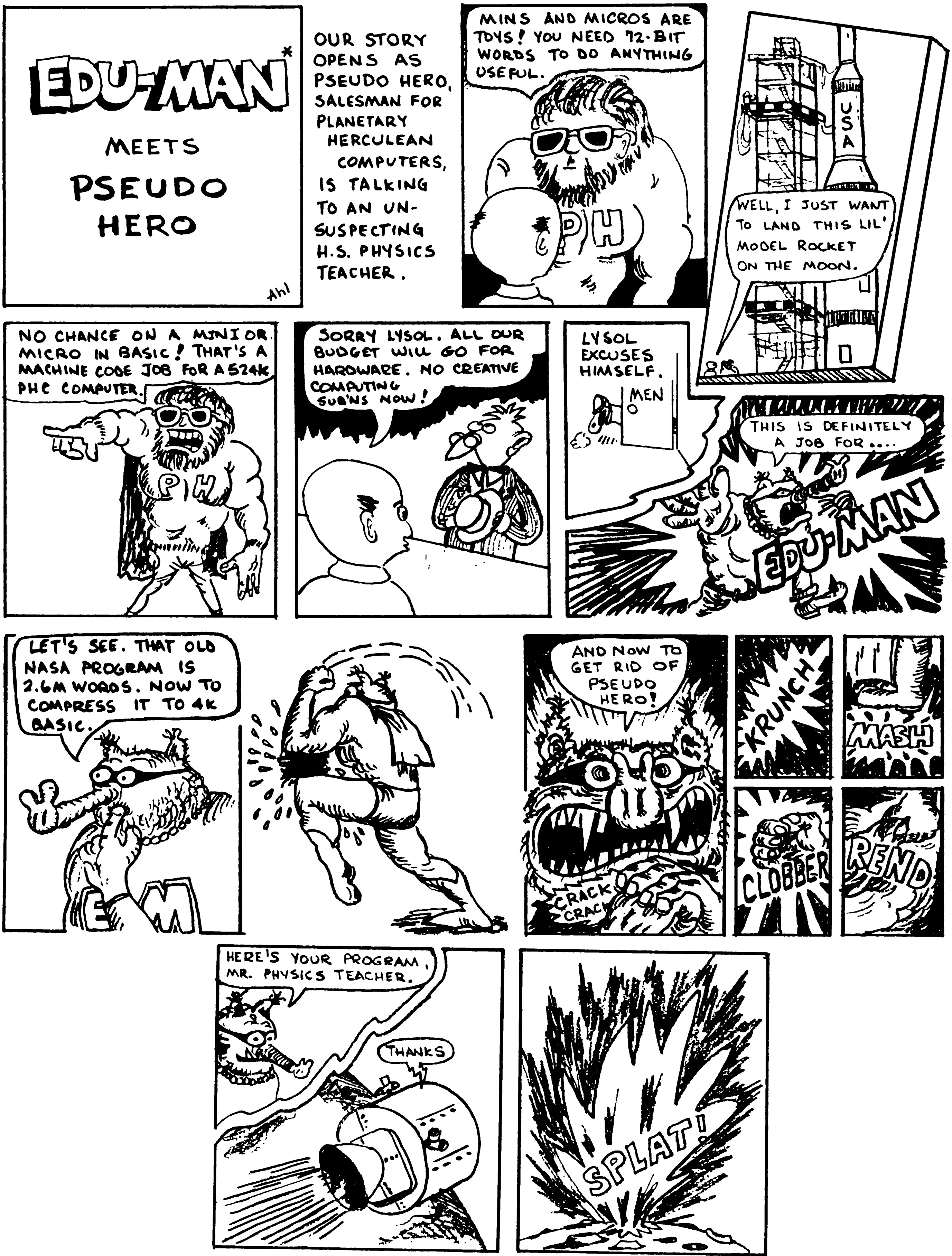
Of course I had fired the bookkeeper right away, so I had to stay up late nights with the programmers helping them to iron out the bugs (it's amazing how quickly I learned the colorful jargon of the trade). The fact that the programs didn't work was really of secondary importance, since we had a much more serious problem in that the data base periodically vanished from the disks. This, as the salesman so helpfully explained, was because of an unfortunate typographical error in the engineering specifications for the disk drives, as a result of which the drives functioned only within a temperature range of two degrees and were also especially sensitive to humidity and to the presence of nylon stockings. We discovered this when a woman programmer unexpectedly became pregnant and quit. The implementation timetable was knocked back by eleven months since she hadn't documented a thing and everything had to be rewritten from scratch, but disk errors were less frequent afterwards.

Once a group of three technicians spent two days replacing all the circuit cards. Things were better for a while, but soon everything was back to the normal state of chaos.

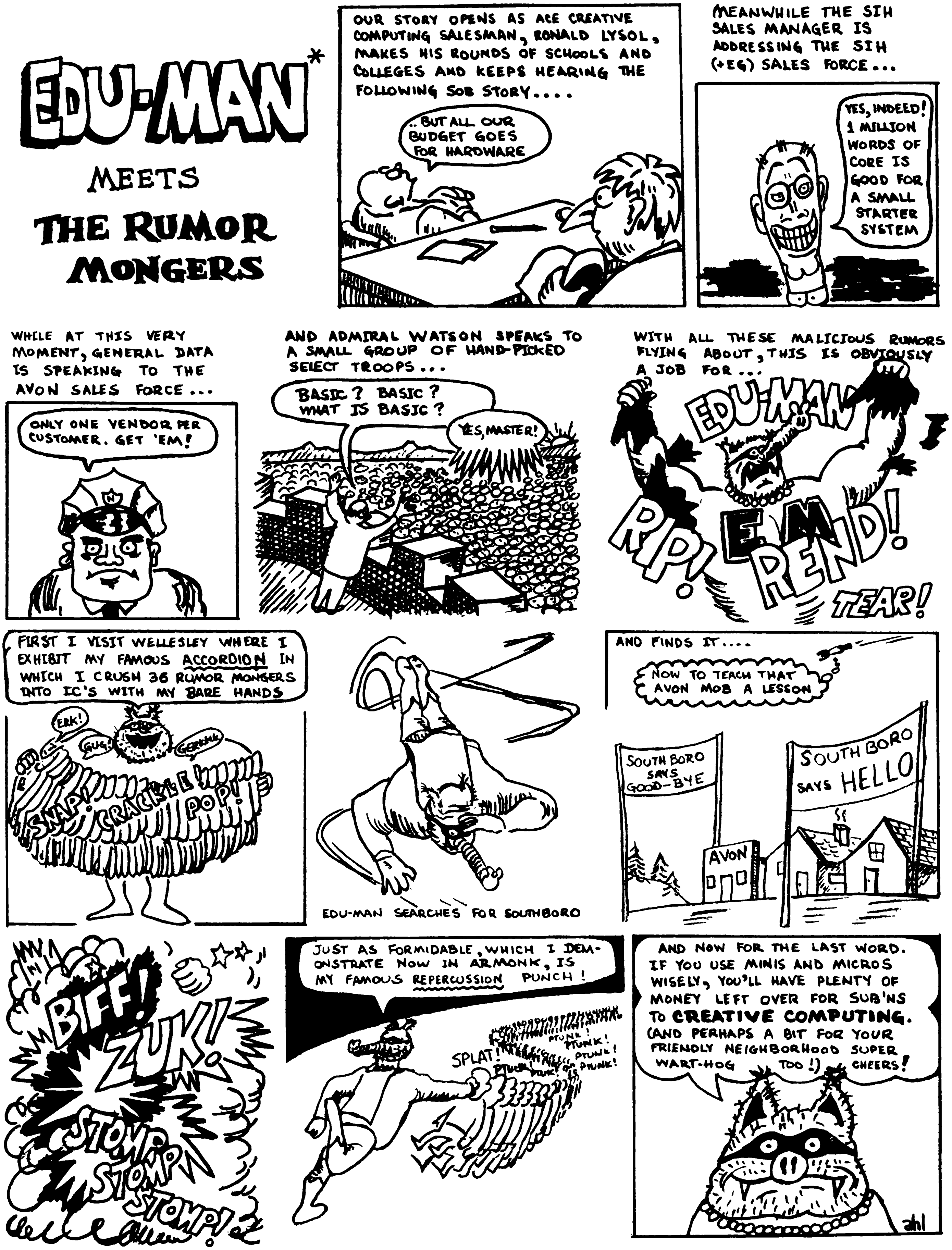
Finally, after a year and a half, enough programs were ready so that we could start relying on the computer and stop triple checking its arithmetic. We had a little party and got drunk, and since then everything has been just fine.

Our salesman has got me thinking about a data base telecommunications on line order entry enquiry system with a little TV terminal in my house. He said the programming effort for the conversion should take about a month, since the company has a new super language. I'm giving the subject some serious thought.

As for the ultimate question of whether we saved any money, it's a bit difficult to say. We did save the salaries of the bookkeeper and some other clerical help, but this was offset by some hardware we had to buy, programming services, orders canceled because of delays directly attributable to the computer's idiosyncracies, and medication for an ulcer I developed. For the year and a half we're ahead by about $40, I think, or about four bits a week. I'm not complaining. It could have been worse. I could still have my wife's second cousin working here. ■


EDU-MAN*
MEETS
PSEUDO
HERO
Ahl
OUR STORY OPENS AS PSEUDO HERO, SALESMAN FOR PLANETARY HERCULEAN COMPUTERS, IS TALKING TO AN UN-SUSPECTING H.S. PHYSICS TEACHER.
MINS AND MICROS ARE TOYS! YOU NEED 72-BIT WORDS TO DO ANYTHING USEFUL.
PH
USA
WELL, I JUST WANT TO LAND THIS LIL' MODEL ROCKET ON THE MOON.
NO CHANCE ON A MINI OR MICRO IN BASIC! THAT'S A MACHINE CODE JOB FOR A 524K PHC COMPUTER.
PH
SORRY LYSOL. ALL OUR BUDGET WILL GO FOR HARDWARE. NO CREATIVE COMPUTING SUB'NS NOW!
LYSOL EXCUSES HIMSELF.
MEN
THIS IS DEFINITELY A JOB FOR....
EDU-MAN
LET'S SEE. THAT OLD NASA PROGRAM IS 2.6M WORDS. NOW TO COMPRESS IT TO 4K BASIC.
EM
AND NOW TO GET RID OF PSEUDO HERO!
CRACK CRACK
KRUNCH
MASH
CLOBBER
REND
HERE'S YOUR PROGRAM, MR. PHYSICS TEACHER.
THANKS
SPLAT!

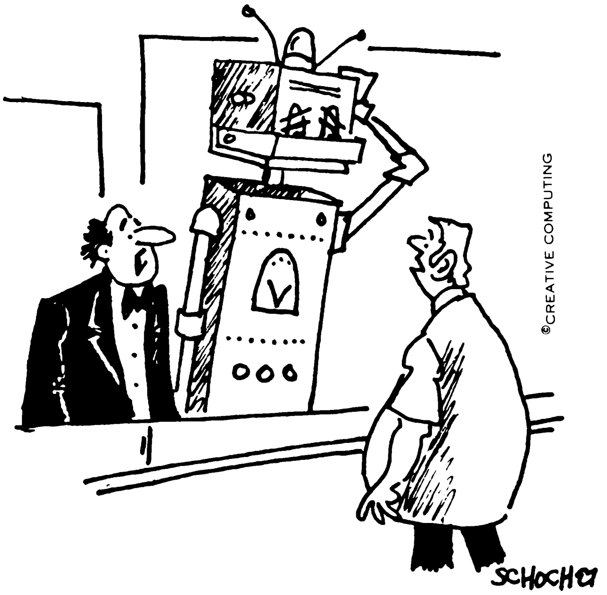
* Yes, astute readers — Creative Computing's erudite, effervescent, easy-going Edu-Man is indeed the twin brother of fearless, fighting, foulmouthed Wonder Wart Hog. Apologies to Gilbert Shelton. — DHA


EDU-MAN*
MEETS
THE RUMOR MONGERS
OUR STORY OPENS AS ACE CREATIVE COMPUTING SALESMAN, RONALD LYSOL, MAKES HIS ROUNDS OF SCHOOLS AND COLLEGES AND KEEPS HEARING THE FOLLOWING SOB STORY....
..BUT ALL OUR BUDGET GOES FOR HARDWARE
MEANWHILE THE SIH SALES MANAGER IS ADDRESSING THE SIH (+EG) SALES FORCE...
YES, INDEED! 1 MILLION WORDS OF CORE IS GOOD FOR A SMALL STARTER SYSTEM
WHILE AT THIS VERY MOMENT, GENERAL DATA IS SPEAKING TO THE AVON SALES FORCE...
ONLY ONE VENDOR PER CUSTOMER. GET 'EM!
AND ADMIRAL WATSON SPEAKS TO A SMALL GROUP OF HAND-PICKED SELECT TROOPS...
BASIC? BASIC? WHAT IS BASIC?
YES, MASTER!
WITH ALL THESE MALICIOUS RUMORS FLYING ABOUT, THIS IS OBVIOUSLY A JOB FOR...
EDU-MAN
RIP!
E M
REND!
TEAR!
FIRST I VISIT WELLESLEY WHERE I EXHIBIT MY FAMOUS ACCORDION IN WHICH I CRUSH 36 RUMOR MONGERS INTO IC'S WITH MY BARE HANDS
ERK!
GUG!
GERRHK
SNAP! CRACKLE! POP!
EDU-MAN SEARCHES FOR SOUTHBORO
AND FINDS IT....
NOW TO TEACH THAT AVON MOB A LESSON
SOUTH BORO SAYS GOOD-BYE
SOUTH BORO SAYS HELLO
AVON
BIFF!
ZUK!
STOMP STOMP STOMP STOMP!
JUST AS FORMIDABLE, WHICH I DEMONSTRATE NOW IN ARMONK, IS MY FAMOUS REPERCUSSION PUNCH!
SPLAT!
PTUNK! PTUNK! PTUNK! PTUNK!
AND NOW FOR THE LAST WORD. IF YOU USE MINIS AND MICROS WISELY, YOU'LL HAVE PLENTY OF MONEY LEFT OVER FOR SUB'NS TO CREATIVE COMPUTING. (AND PERHAPS A BIT FOR YOUR FRIENDLY NEIGHBORHOOD SUPER WART-HOG TOO!) CHEERS!
ahl

* Yes, astute readers — Creative Computing's erudite, effervescent, easy-going Edu-Man is indeed the twin brother of fearless, fighting, foulmouthed Wonder Wart Hog. Apologies to Gilbert Shelton. — DHA

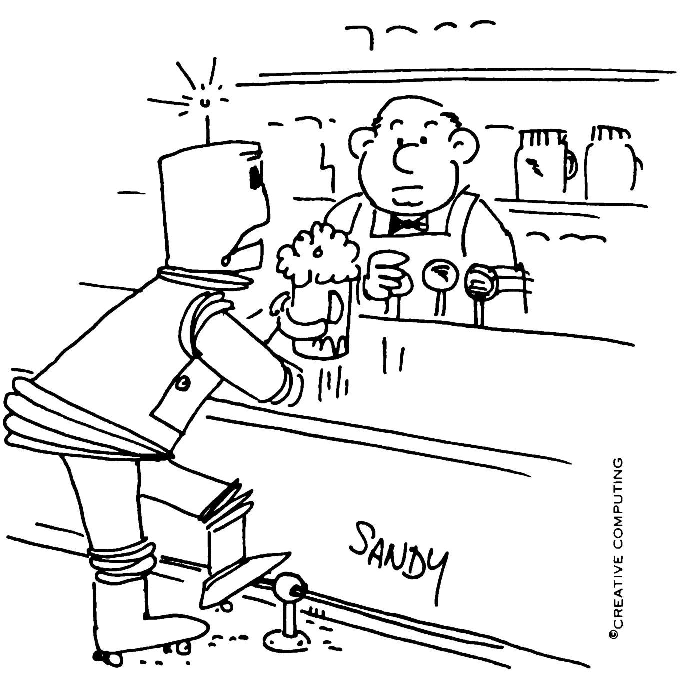
© CREATIVE COMPUTING

THE LIGHTER SIDE OF ROBOTS



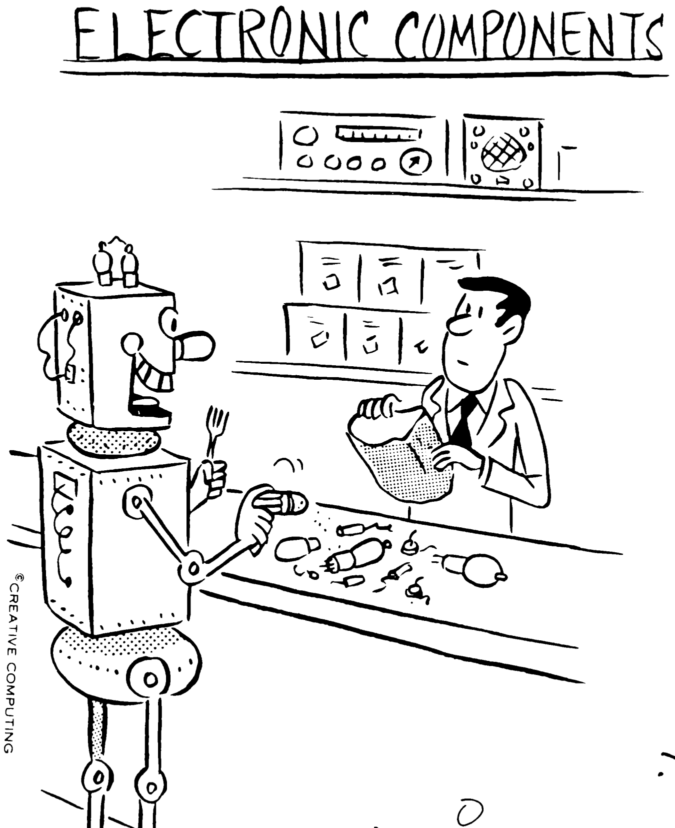
©CREATIVE COMPUTING

"A box of CMOS please. He gets terrible migranes when he has to do intricate figuring."



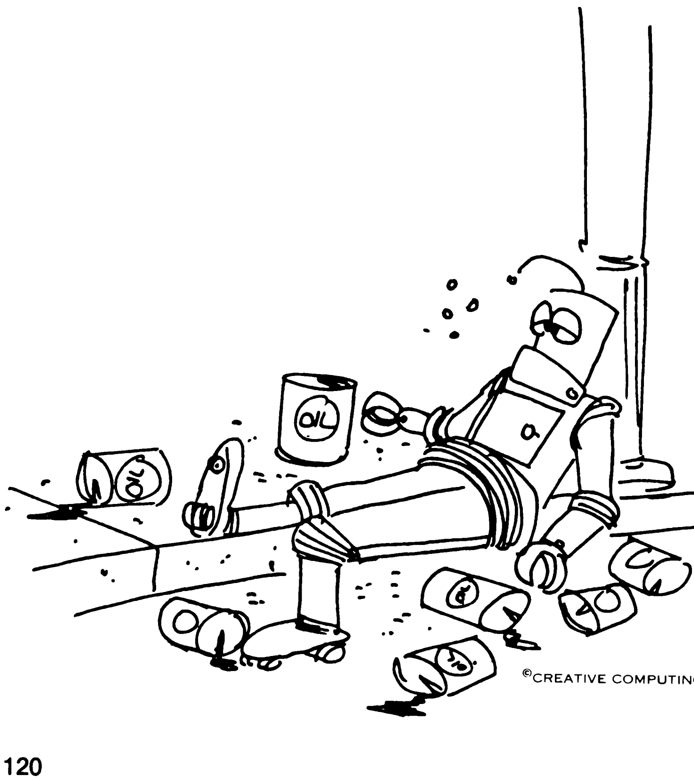
©CREATIVE COMPUTING

"My programmer doesn't understand me."


ELECTRONIC COMPONENTS
Marvin TOWNSEND

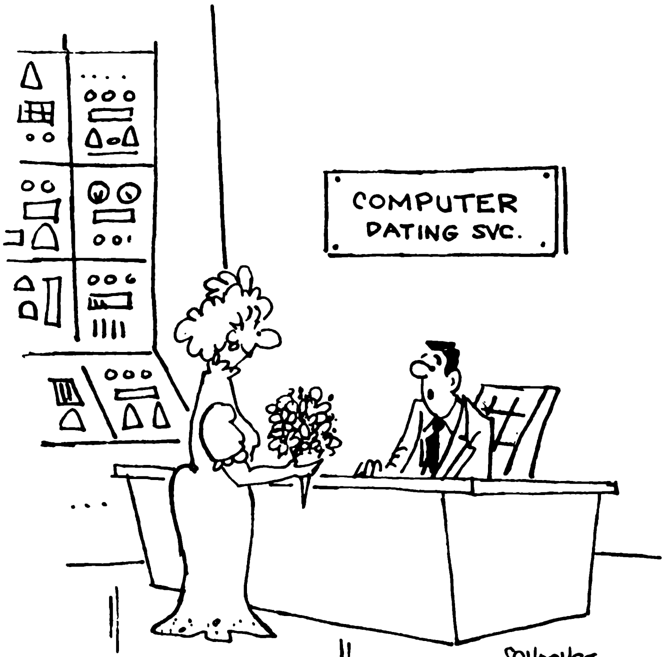
©CREATIVE COMPUTING

"Never mind the sack; I'll eat'em here."



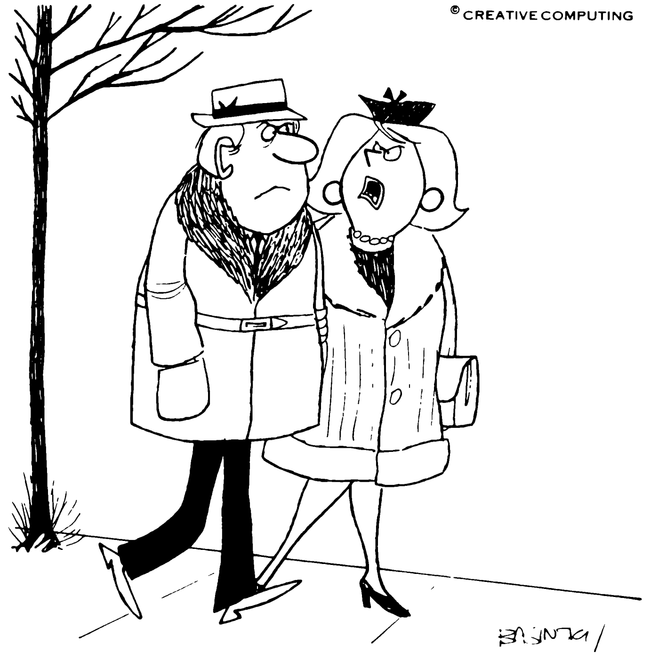
©CREATIVE COMPUTING

THE LIGHTER SIDE OF COMPUTER DATING


COMPUTER
DATING SVC.
SCHOCHET

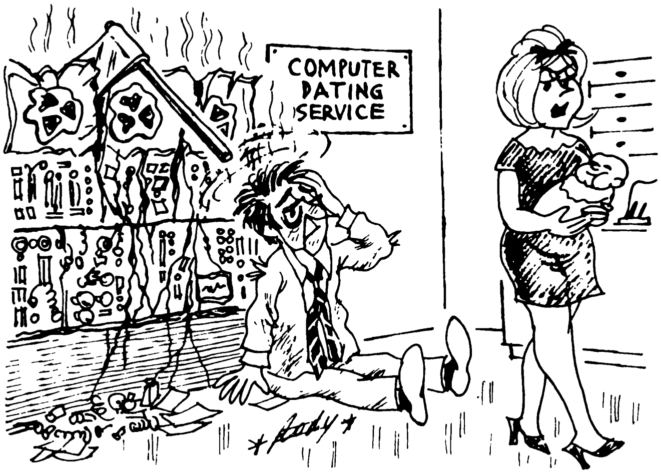
© CREATIVE COMPUTING

"Let's not rush things now Miss Jablonski."

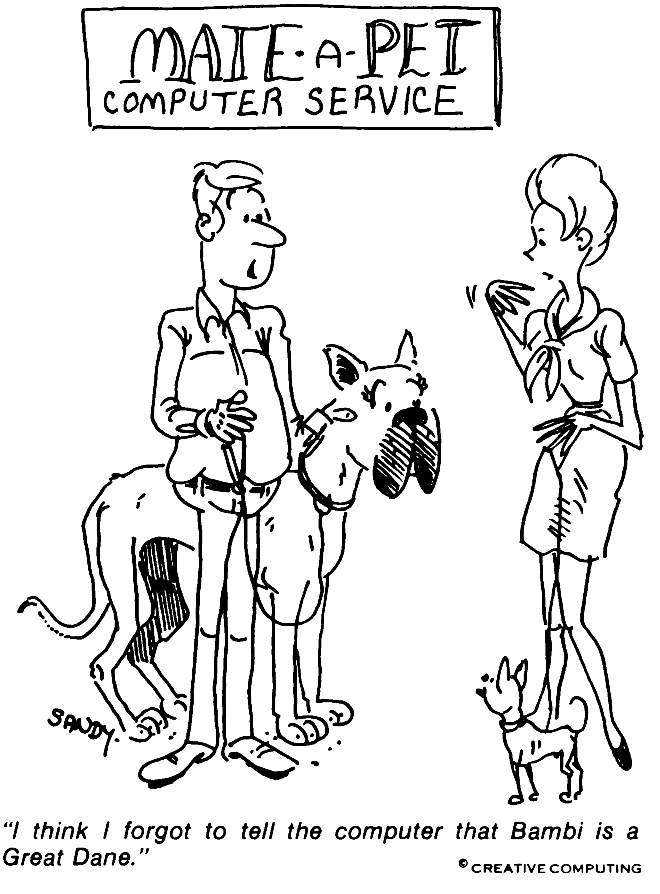


© CREATIVE COMPUTING

"To think I needed a computer to find YOU!"

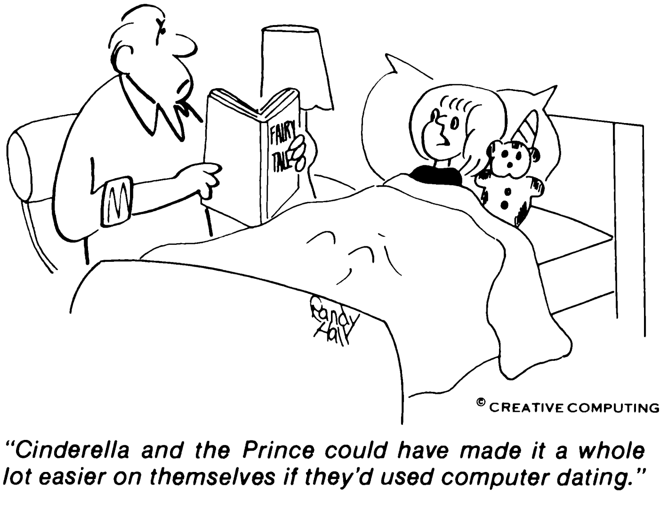

COMPUTER
DATING
SERVICE

© CREATIVE COMPUTING


MATE-A-PET
COMPUTER SERVICE
SANDY

"I think I forgot to tell the computer that Bambi is a Great Dane."

© CREATIVE COMPUTING

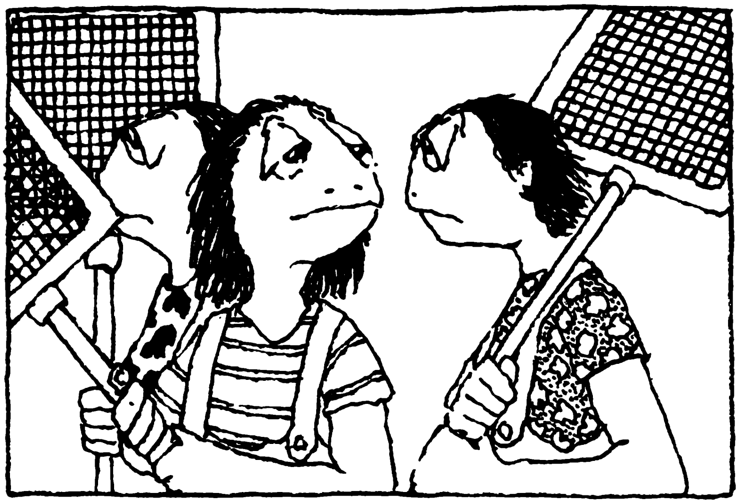

FAIRY TALES
Randy

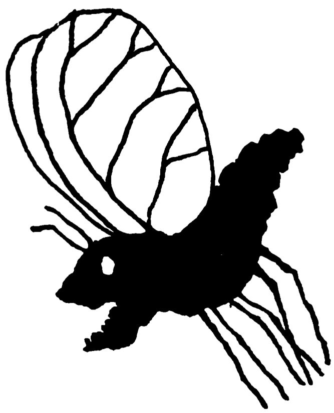
© CREATIVE COMPUTING

"Cinderella and the Prince could have made it a whole lot easier on themselves if they'd used computer dating."

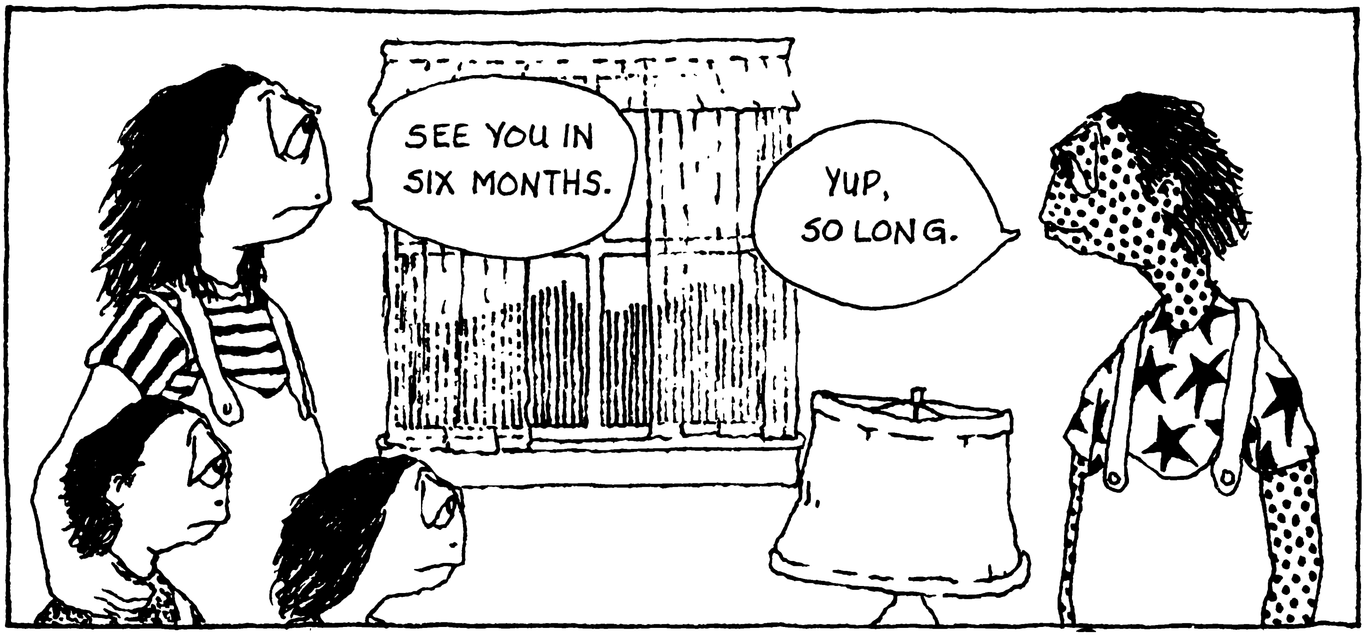
NORDS

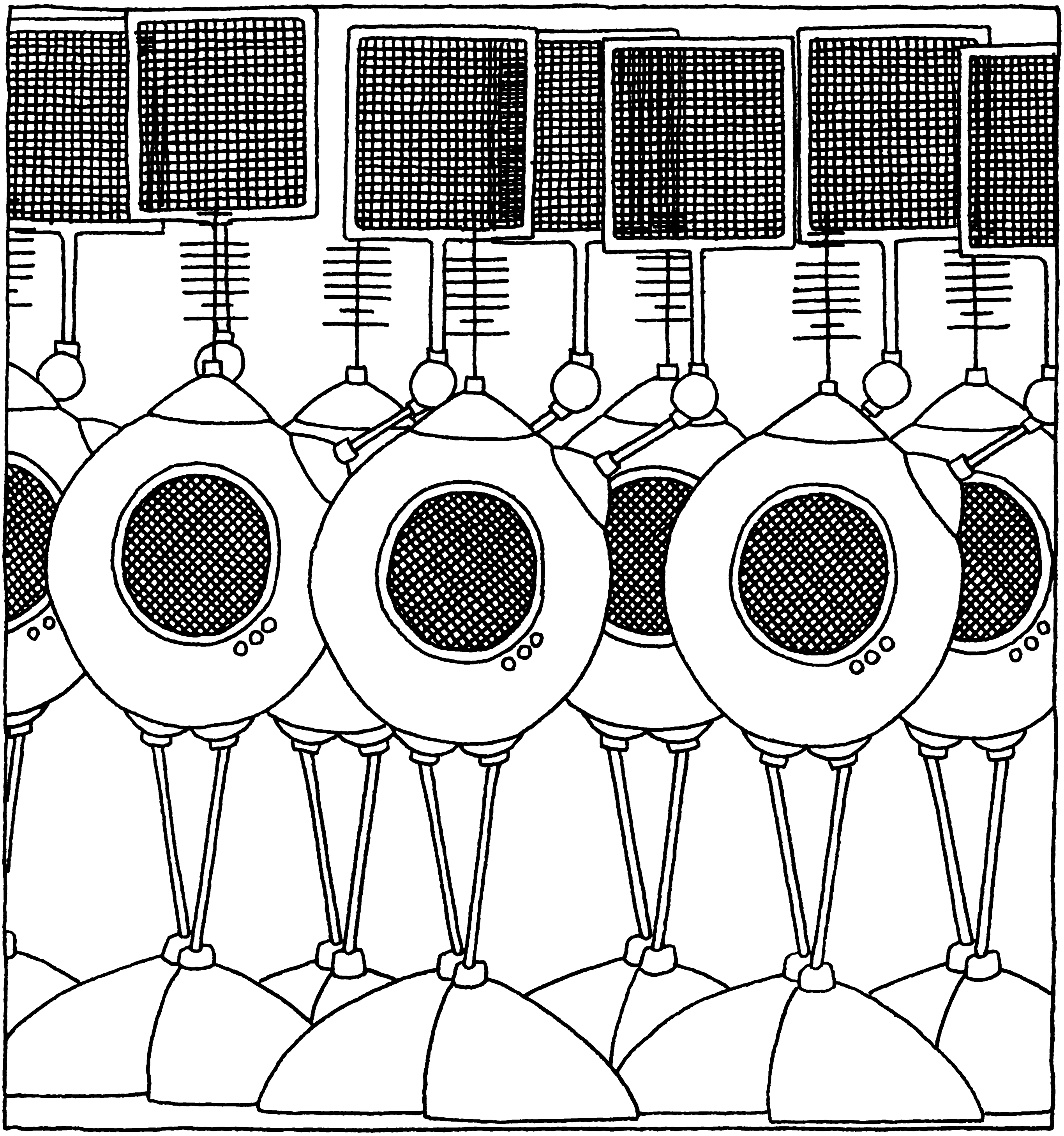
The Nords walked around with their eyes on the sky and their hands on large swatters. There was a good reason for this: the skies were full of large bugs called snits.



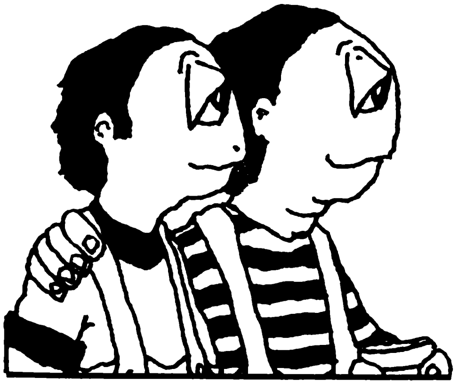


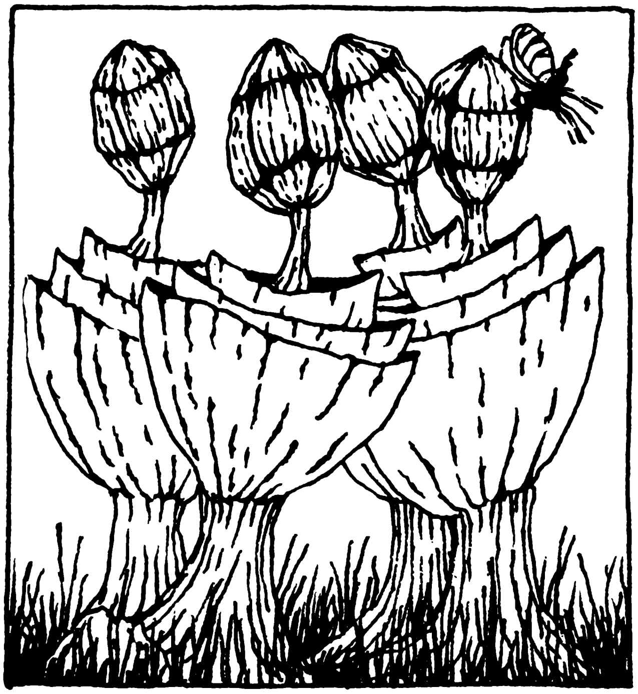
The snits were not considered a blessing. The snits had a bite that felt like the caress of a chain saw. A Nord bitten by a snit broke out in spots and developed an odor so horrendous that it was called The Curse. The condition lasted six months. It was six months in isolation for no one could stand the odor.


SEE YOU IN SIX MONTHS.
YUP, SO LONG.

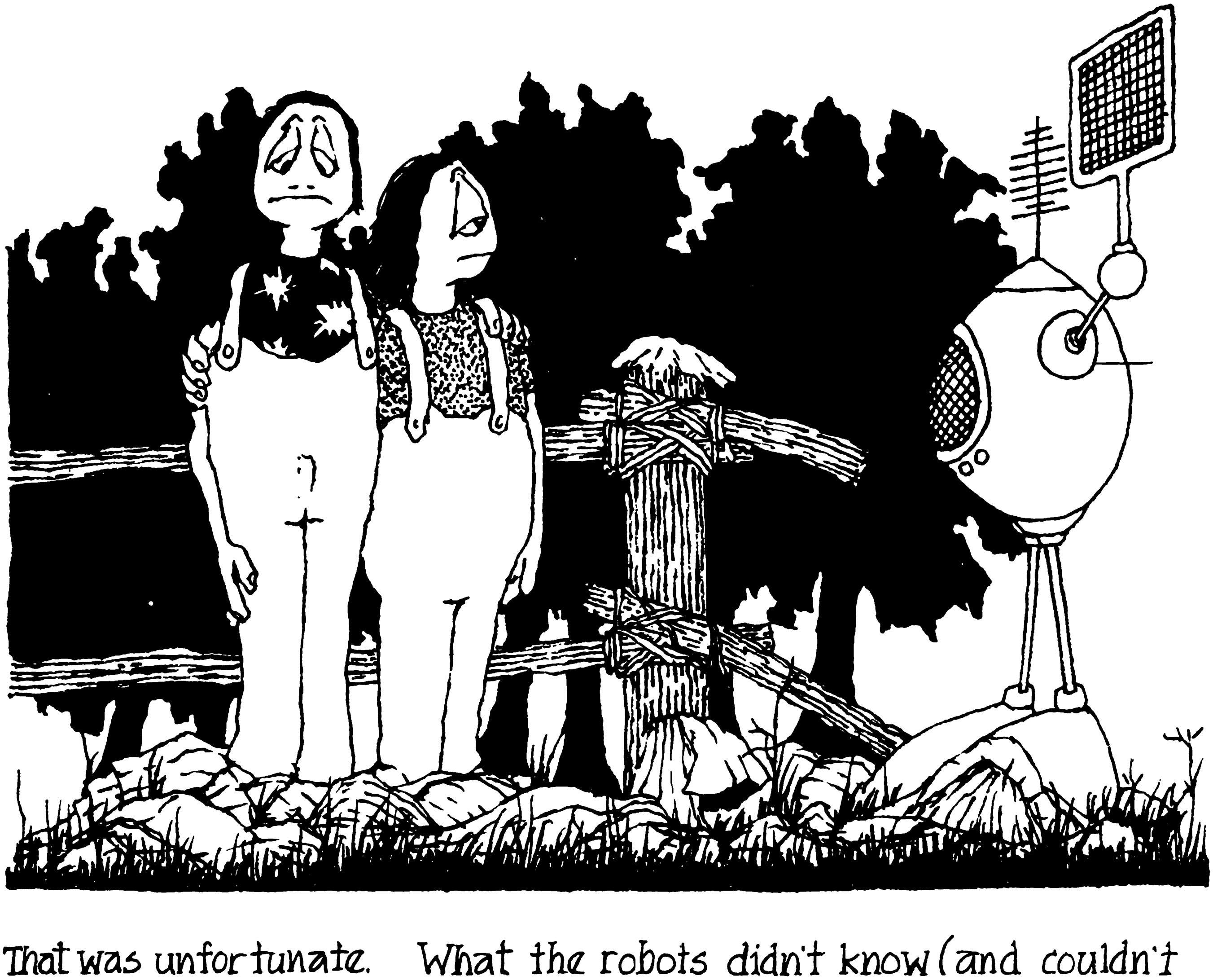


Then the Nords developed the technology to rid themselves of the snits. It was in the form of a robot equipped with a large swatter. The Nords built an army of them and programmed them to kill all snits. The Nords figured heaven was just around the corner.

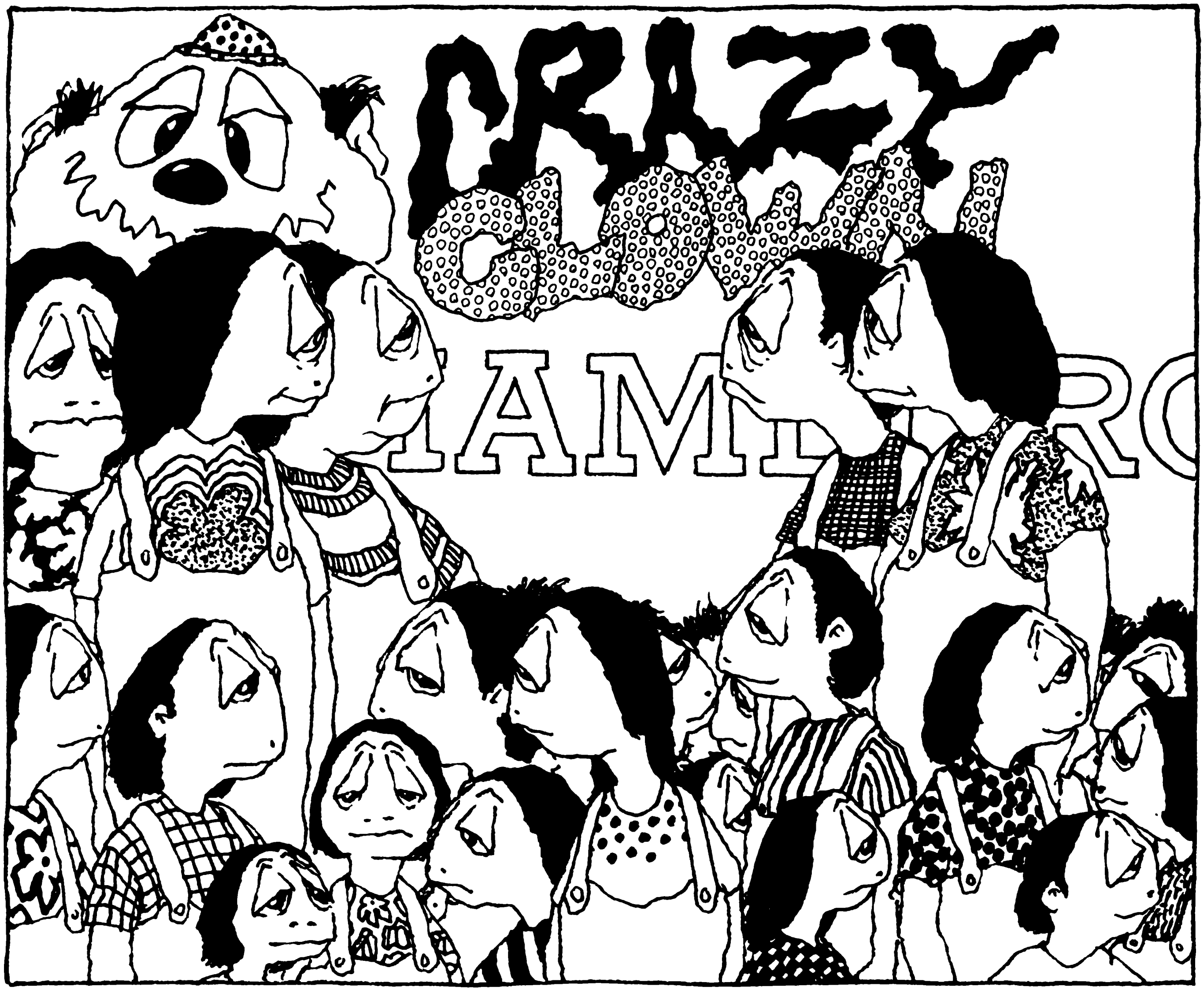




The robots began by chasing snits all over creation but it wasn't long before the robots discovered that the snits had only one source of food: the leaves of the Maunt plants. Programmed to destroy all snits and computing cause and effect as being one and the same, the robots used their swatters to reduce all the Maunts to mangled pulp.

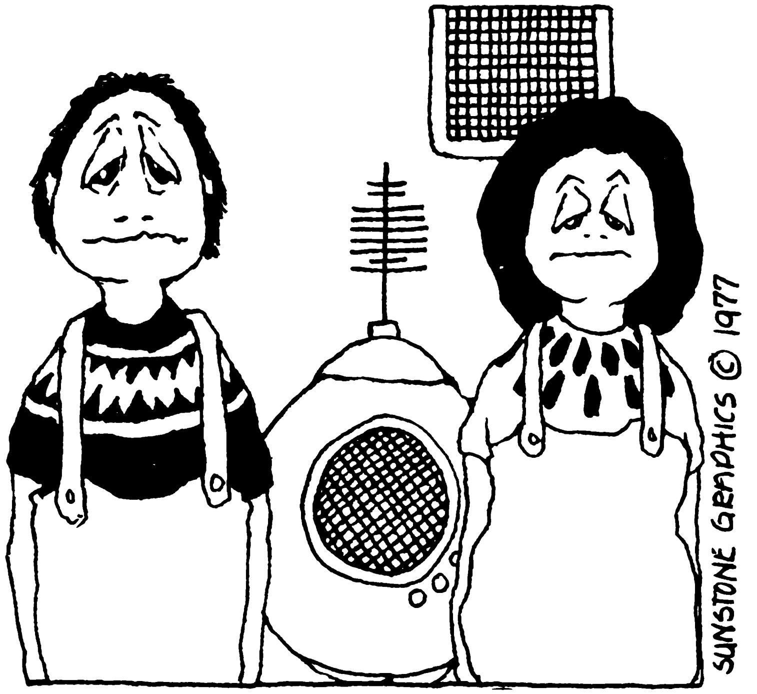


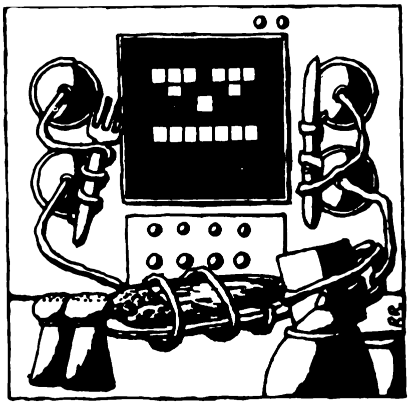
That was unfortunate. What the robots didn't know (and couldn't have known since the information wasn't included in their programs) was that the seeds of the Maunts were the only source of birth control the Nords had.


CRAZY
CLOWN
AMD RO

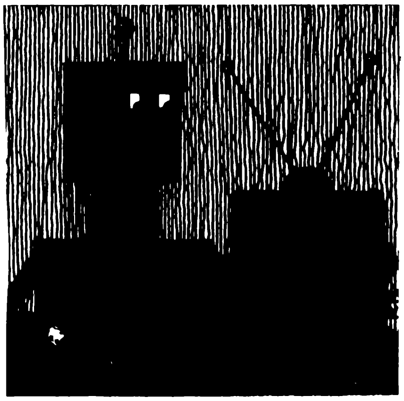
The population increase was immediate and immense.

Desperation set in. Laws were passed. Mating was restricted to three days a year. The robots were reprogrammed and used as police. One was assigned to each home. Any couple attempting to mate on any other than the three designated days faced the possibility of being reduced to mangled pulp.


SUNSTONE GRAPHICS © 1977



A computer invited to sup,
fumbled with platter and cup,
with fork and with knife,
and, asked about life,
mumbled, "It doesn't add up!"

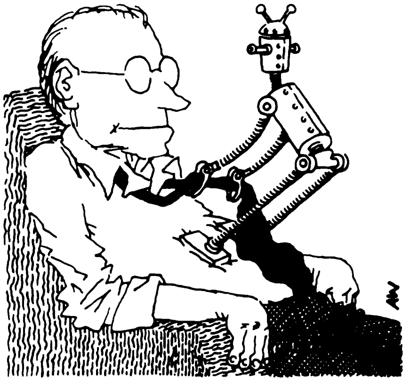


GLOROBOTS

Gloria Maxson

A robot of amorous renown
broke the whole powerplant down
by starting to pet
with an old TV set,
which totally blacked out the town.

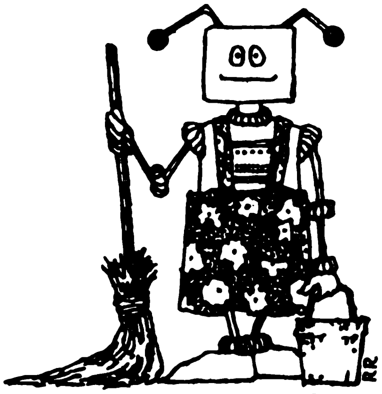
There'll be computers of size,
and some they will miniaturize
to do the small tasks
that everyone asks,
from scratching, to knotting of ties.



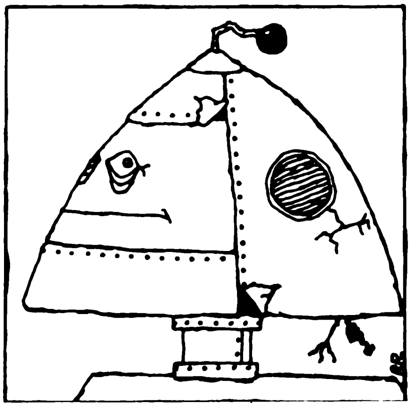
A new synthesizer named Moog,
and an old-fashioned organ named Oog,
had a concert to test
which one played best,
and are having it yet, Moog and Oog.

I heard an old robot explain:
"We were evolved to be sane,
with tolerant views,
so men need not use
the other nine-tenths of their brain."

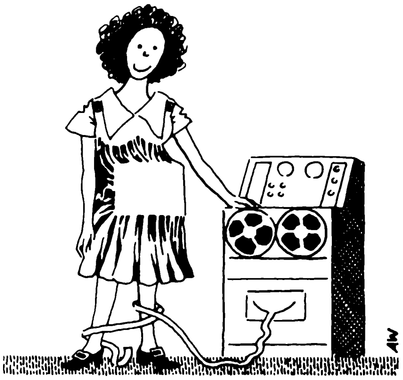
A robot with lofty inflection
read Stein in the poetry section,
but read it, "Arose
is arose is arose,"
and thought it concerned resurrection.



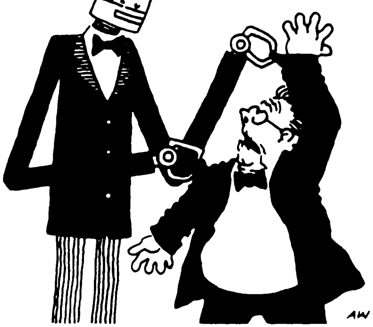
A robot domestic named Grace
would work any time, any place,
and never relent,
because she was bent
on expanding the customer base.



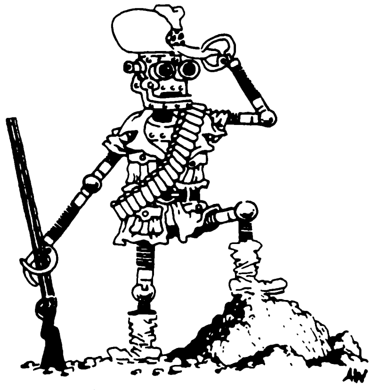
An old robot thought it was hell
that he was unable to tell
how feeble he got,
since it was his lot
to be terminal even when well.



A computer designed to compute
a couple's rapport or dispute,
typed "Yes!" unashamed
when the young lady named
it in her paternity suit.



A robotical butler named Jeeves
was given no further reprieves
after asking a guest
for his coat, hat — and vest,
shoes and socks, shorts, and shirtsleeves.



A robot in tracking had fun
in a way calculated to stun:
he wore hunting suits,
bold hats, and big boots,
and carried an elephant gun.

The Floating Point Solution

"Delete it!"

"....it has to be a hardware error..."

David Ahl
CREATIVE COMPUTING

Dear Mr. Ahl:

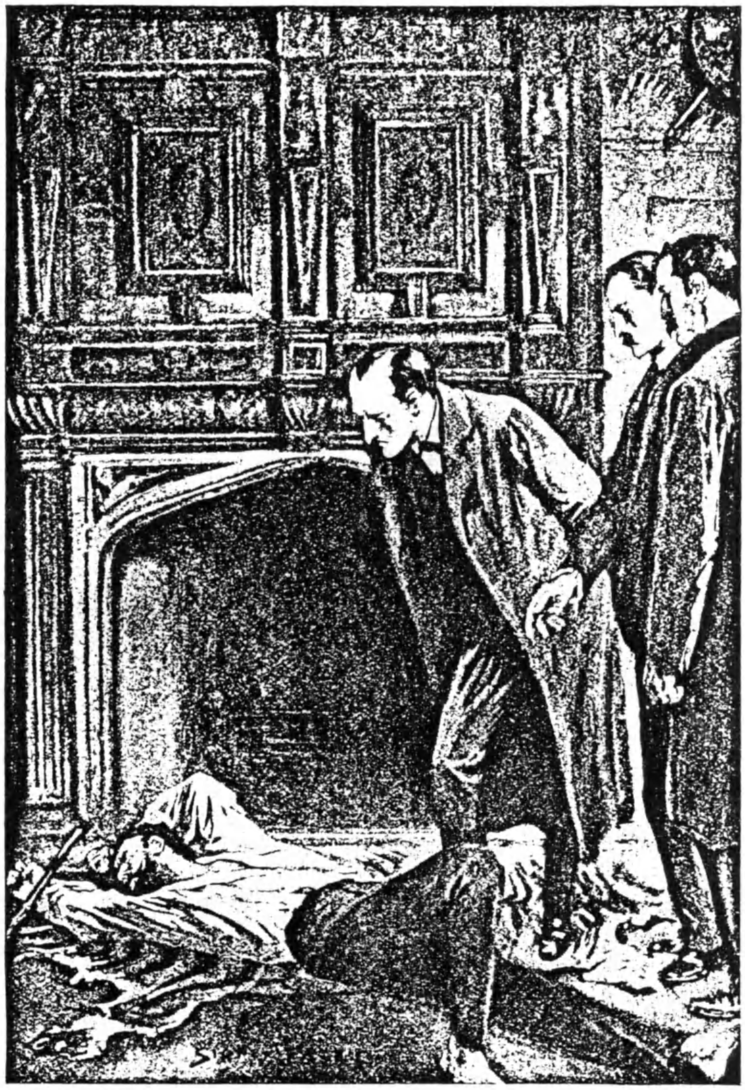
Nicholas Meyer discovered the previously unknown Sherlock Holmes adventure, *The Seven-Per-Cent Solution* and electrified Holmes's present day followers with the establishment of a link between Holmes and Freud. I have just discovered a link between Holmes and several of the pioneer thinkers in computing, an exciting tale which Watson appropriately dubbed *The Floating Point Solution.* Certainly there are descriptions of Holmes in many of the previously published tales which suggest how close his mentality was to that of the typical programmer/analyst. Nothing, though, even approaches the truth embodied in *The Floating Point Solution.*

Unfortunately, the manuscript itself is not yet in a publishable form and so I can not really submit it to you for serialization in *Creative Computing.* At this time, the most I can do is allow you to publish a few of the Sidney Paget illustrations which had been prepared for this tale but were never published. The captions will convey a bit of the drama of the tale itself but will not, I trust, give the story away completely.

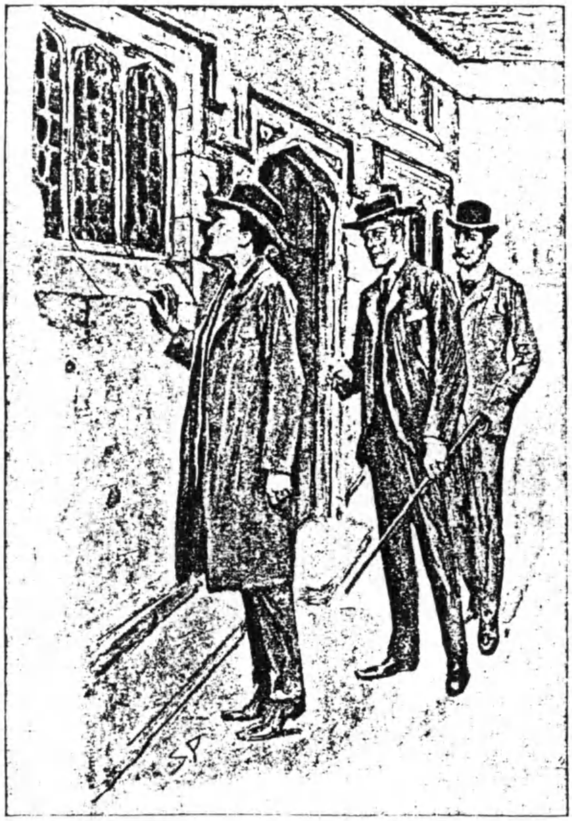
You may expect to hear from me again.

Sincerely yours,

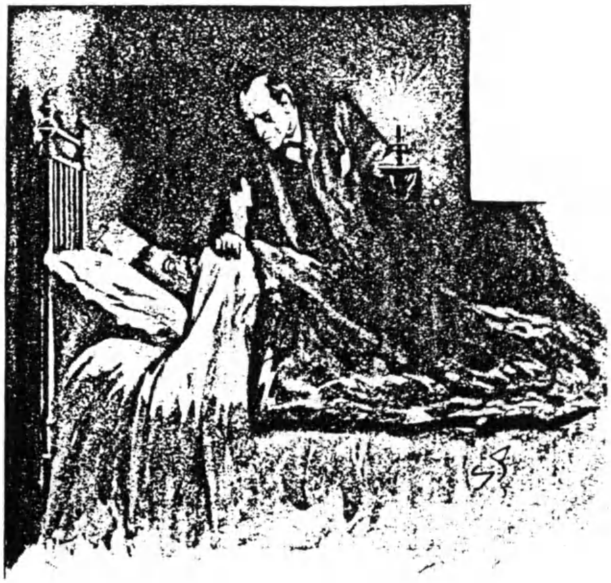
Robert P. Taylor, Esq.
Kings College, Budo



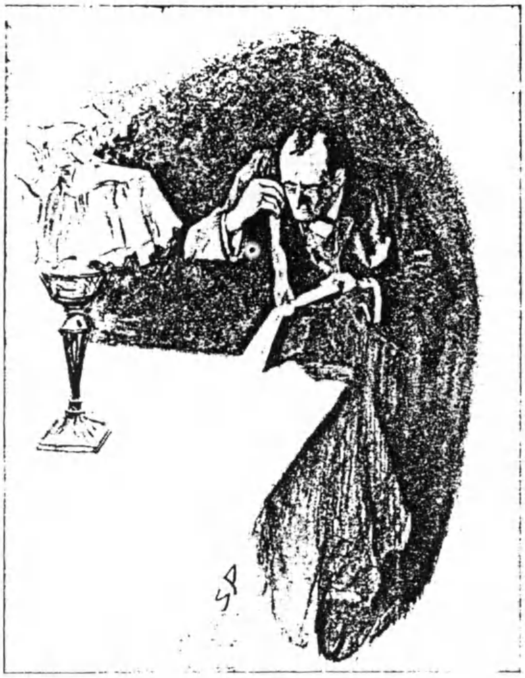
"Yes," mused Holmes, pondering the unfortunate operator's body, "one never knows in advance how the disgrúntled user may react......"



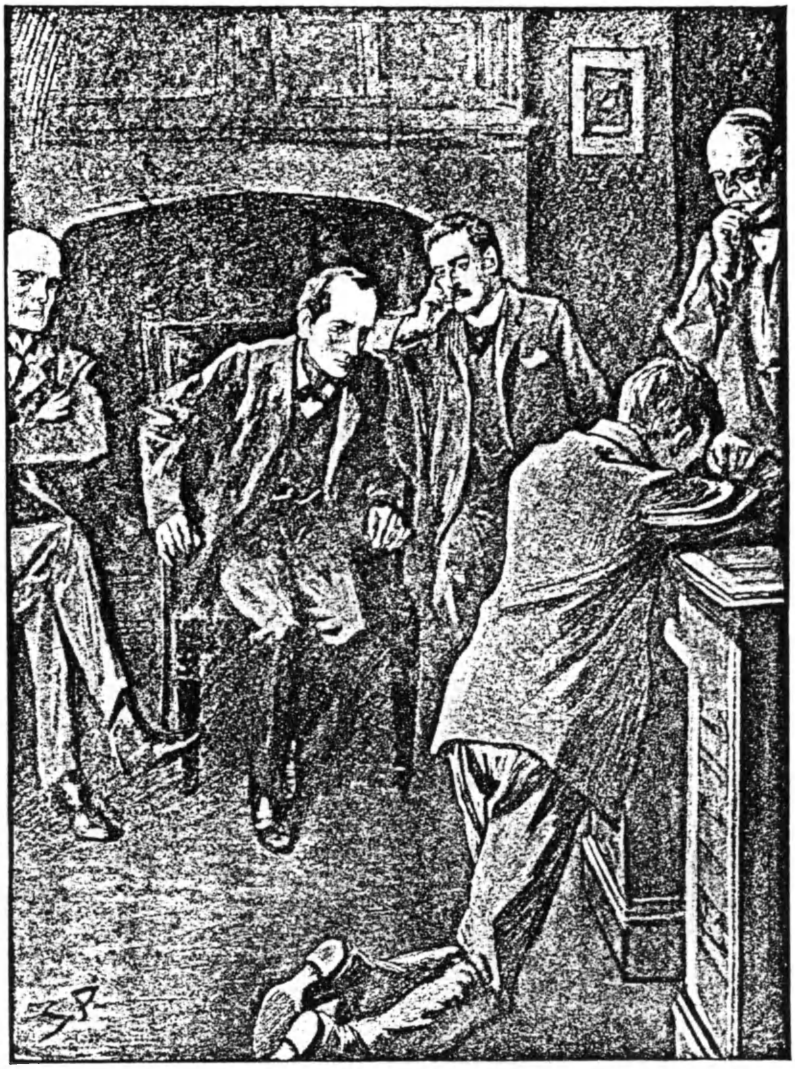
"My guess, Watson, is that it's some sort of artificial brain..."



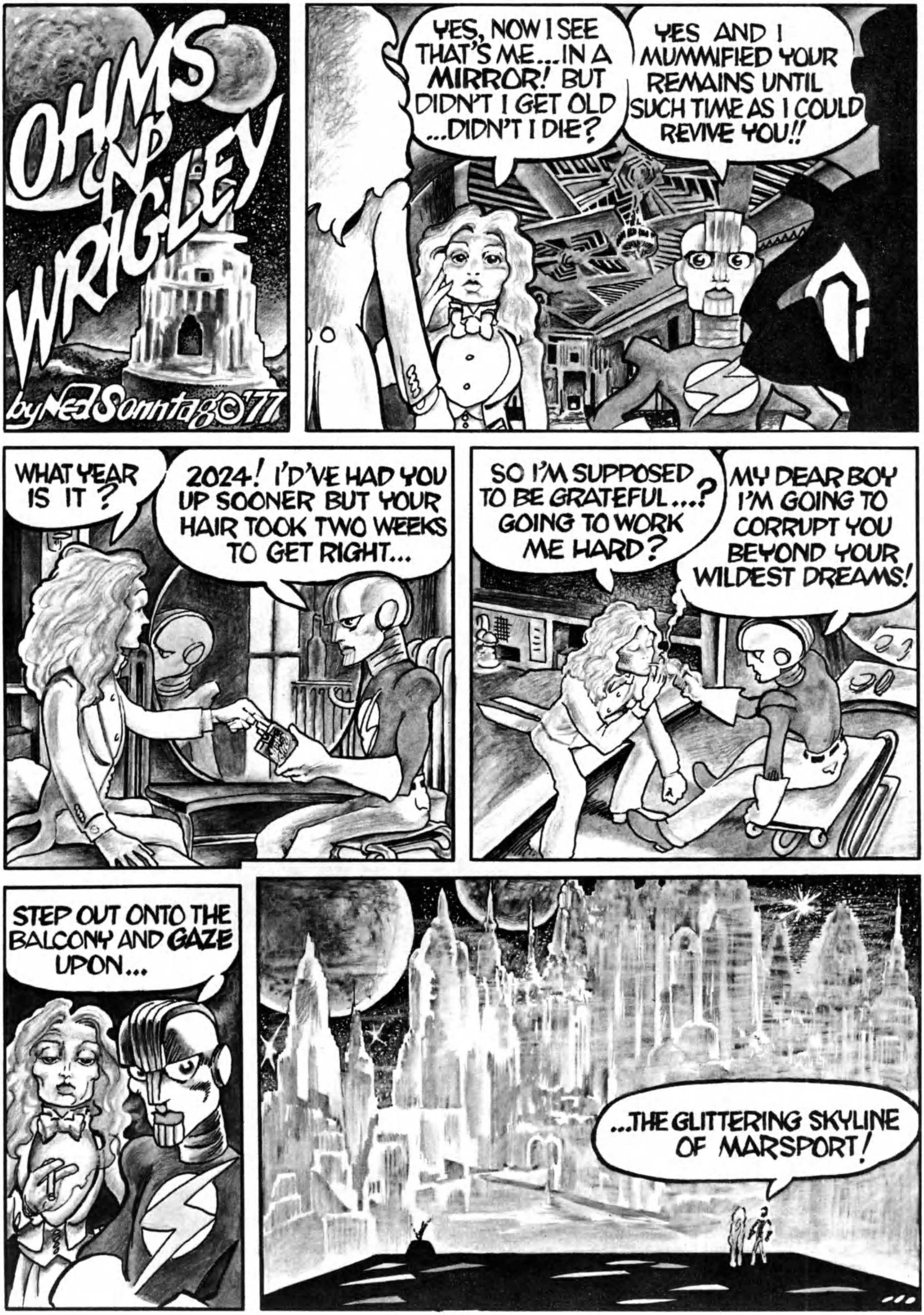
"Hurry, Watson!" cried Holmes. "This is no accidental power loss."



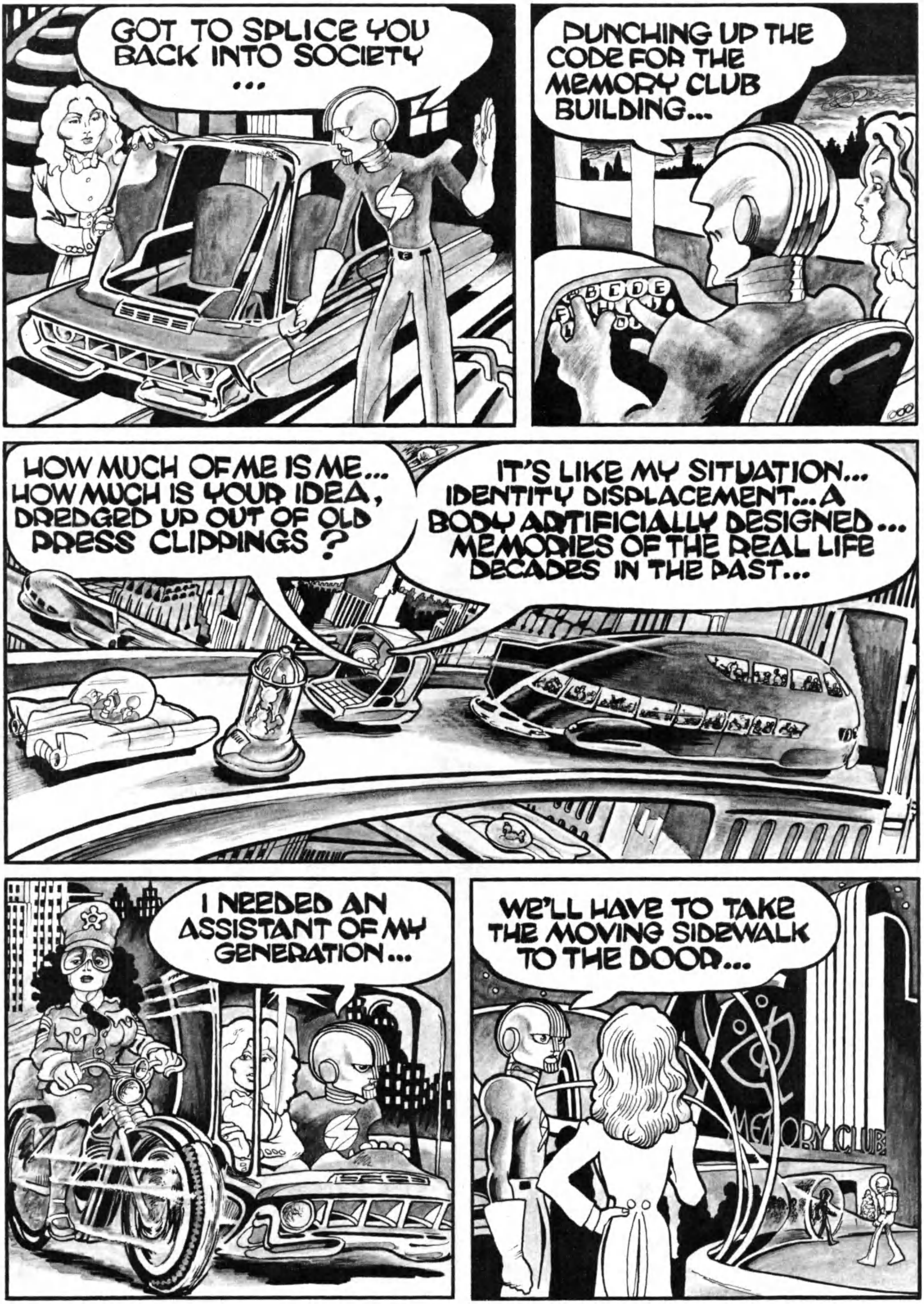
"It must be secret code," thought Holmes, "....it certainly doesn't make sense as a users manual...."

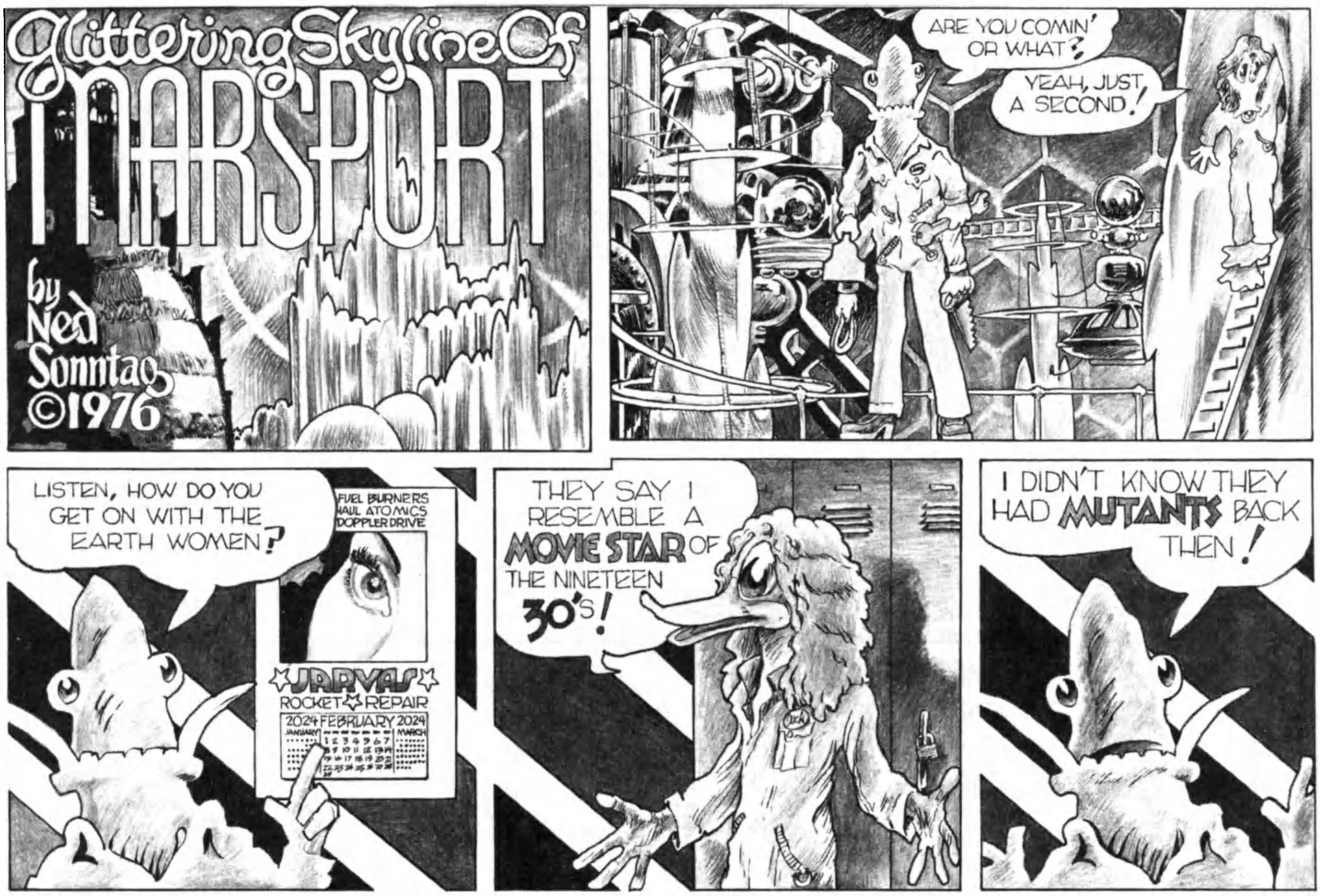


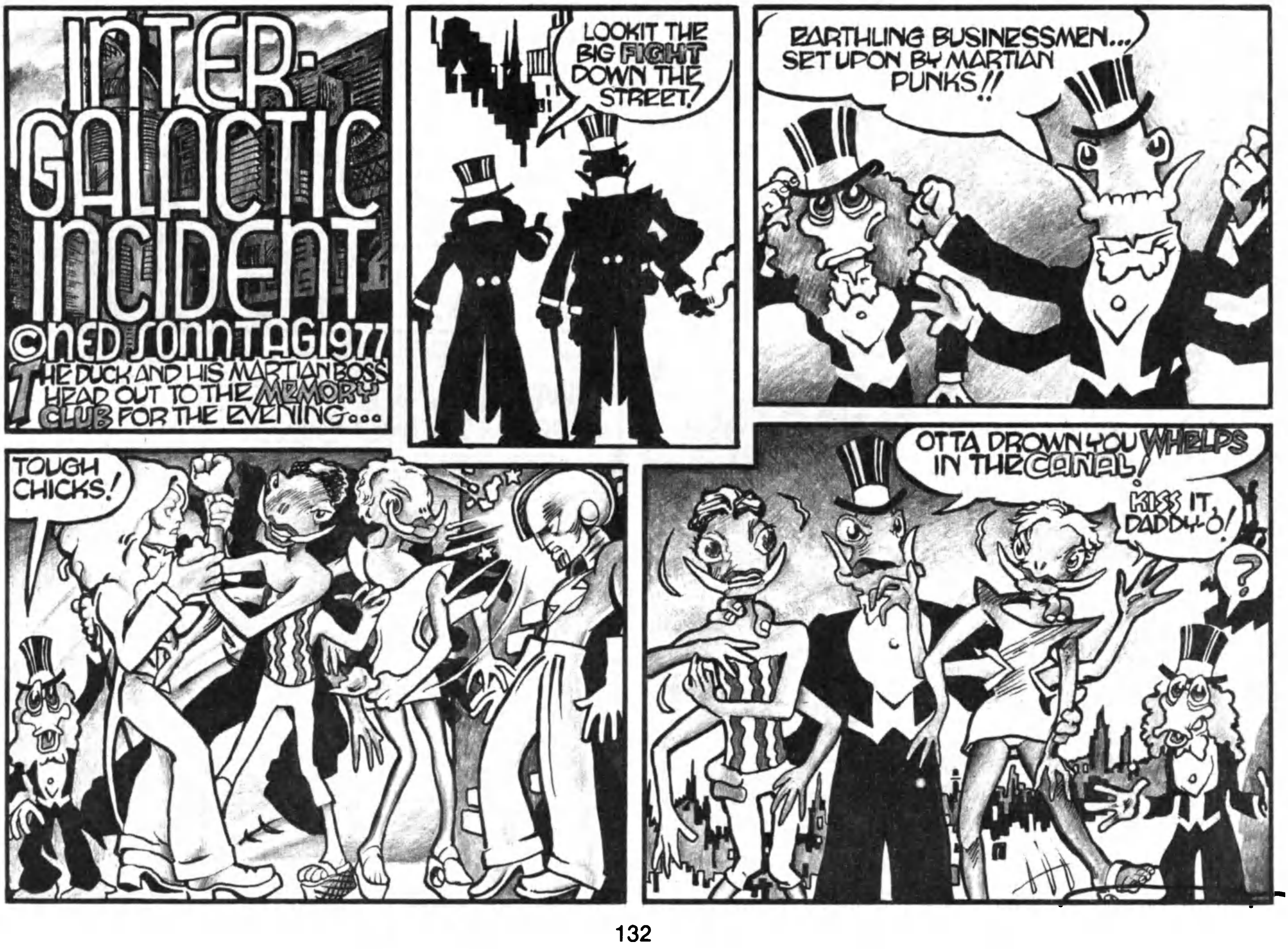
"Come, come," said Holmes, kindly, "at least you still have the listings."

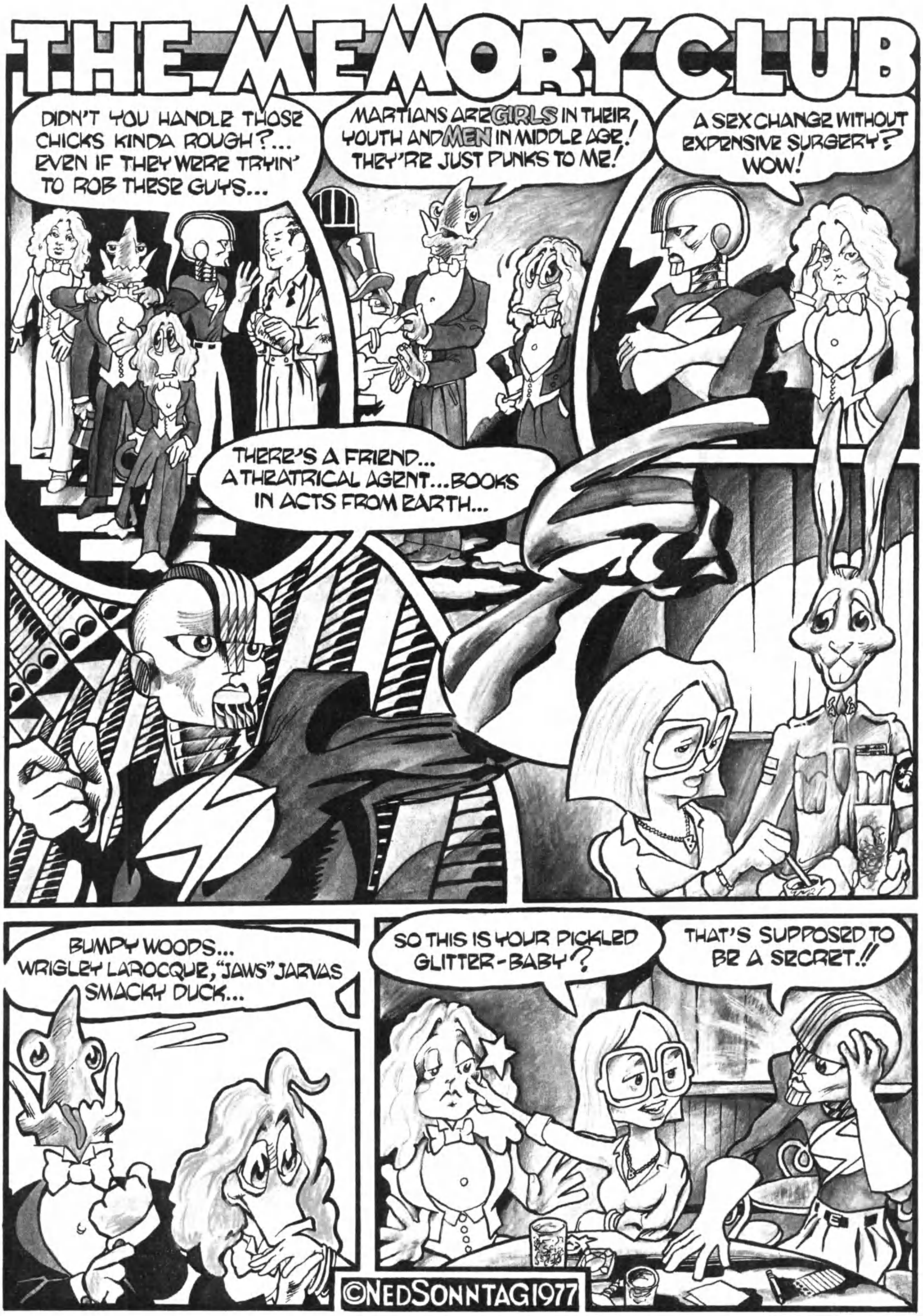

OHMS AND WRIGLEY
by Ned Sonntag ©'77
YES, NOW I SEE THAT'S ME...IN A MIRROR! BUT DIDN'T I GET OLD ...DIDN'T I DIE?
YES AND I MUMMIFIED YOUR REMAINS UNTIL SUCH TIME AS I COULD REVIVE YOU!!
WHAT YEAR IS IT?
2024! I'D'VE HAD YOU UP SOONER BUT YOUR HAIR TOOK TWO WEEKS TO GET RIGHT...
SO I'M SUPPOSED TO BE GRATEFUL...? GOING TO WORK ME HARD?
MY DEAR BOY I'M GOING TO CORRUPT YOU BEYOND YOUR WILDEST DREAMS!
STEP OUT ONTO THE BALCONY AND GAZE UPON...
...THE GLITTERING SKYLINE OF MARSPORT!

© 1977 Ned Sonntag


GOT TO SPLICE YOU BACK INTO SOCIETY ...
PUNCHING UP THE CODE FOR THE MEMORY CLUB BUILDING...
HOW MUCH OF ME IS ME... HOW MUCH IS YOUR IDEA, DREDGED UP OUT OF OLD PRESS CLIPPINGS ?
IT'S LIKE MY SITUATION... IDENTITY DISPLACEMENT... A BODY ARTIFICIALLY DESIGNED... MEMORIES OF THE REAL LIFE DECADES IN THE PAST...
I NEEDED AN ASSISTANT OF MY GENERATION...
WE'LL HAVE TO TAKE THE MOVING SIDEWALK TO THE DOOR...
MEMORY CLUB


Glittering Skyline Of
MARSPORT
by Ned Sonntag ©1976
ARE YOU COMIN' OR WHAT?
YEAH, JUST A SECOND!
LISTEN, HOW DO YOU GET ON WITH THE EARTH WOMEN?
FUEL BURNERS
HAUL ATOMICS
DOPPLER DRIVE
JARVAS ROCKET REPAIR
2024 FEBRUARY 2024
JANUARY
MARCH
THEY SAY I RESEMBLE A MOVIE STAR OF THE NINETEEN 30'S!
I DIDN'T KNOW THEY HAD MUTANTS BACK THEN!

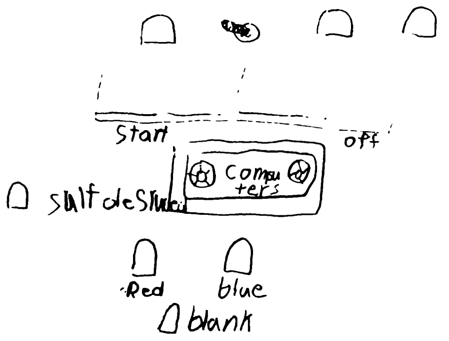

INTER-GALACTIC INCIDENT
©NED SONNTAG 1977
THE DUCK AND HIS MARTIAN BOSS HEAD OUT TO THE MEMORY CLUB FOR THE EVENING...
LOOKIT THE BIG FIGHT DOWN THE STREET!
EARTHLING BUSINESSMEN... SET UPON BY MARTIAN PUNKS!!
TOUGH CHICKS!
OTTA DROWN YOU WHELPS IN THE CANAL!
KISS IT, DADDY-O!
?


THE MEMORY CLUB
DIDN'T YOU HANDLE THOSE CHICKS KINDA ROUGH?... EVEN IF THEY WERE TRYIN' TO ROB THESE GUYS...
MARTIANS ARE GIRLS IN THEIR YOUTH AND MEN IN MIDDLE AGE! THEY'RE JUST PUNKS TO ME!
A SEX CHANGE WITHOUT EXPENSIVE SURGERY? WOW!
THERE'S A FRIEND... A THEATRICAL AGENT...BOOKS IN ACTS FROM EARTH...
BUMPY WOODS... WRIGLEY LAROCQUE, "JAWS" JARVAS SMACKY DUCK...
SO THIS IS YOUR PICKLED GLITTER-BABY?
THAT'S SUPPOSED TO BE A SECRET.!!
©NEDSONNTAG1977

"This year's social studies project," I ingeniously announced to the moppets in my elementary class, "is to learn about the computer industry." I proceeded to produce films, books, magazine articles, lectures and the obligatory field trip.

When the term had come to its close, they demonstrated the wealth of their new knowledge in test papers, reports, and homework assignments.

If you are overworked, discouraged or tired, have no fear, the next generation is almost ready, as you will quickly ascertain from these quotes from their writings.



"Take a good long look at a computer. Does it have input, output, a bit of binary? No, you say to all these questions? Then you are not taking a good long look at a computer."

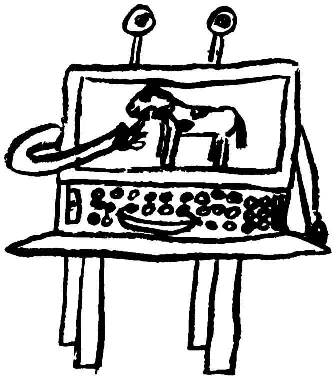
Out of the Mouths of Babes

Eve R. Wirth

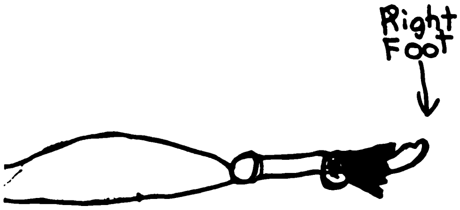
"Girl computer workers have to make real certain all the holes are in the right spots, because if not then how will the computerman get in them."

"Lots of people are working in the hustling bustling business of computers. There could easily be 1000 of them doing this. Maybe 5000 doing this. Might even be a million. I can't be too for sure because it takes just about all of my knowing to even know that lots of people are working in the hustling bustling business of computers."

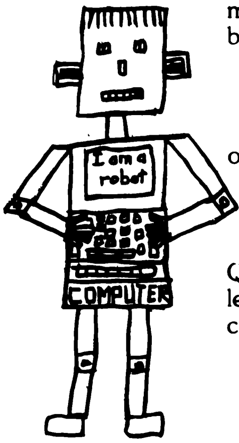
"Just yesterday when I read my libury book I knew real good what computers axaly do for us, but today it's a different colored horse story."



"Question: What do you think is the greatest feat of the invention of the computer? Answer: "I didn't think it has feat, but when I think about it I would say the right foot is the strongest and greatest."



"In the pre-me times of history one day a guy decided to make a machine that could do stuff faster. He thought it was high time for action, so here's what he did. He put alot of holes in it and lots of buttons and stuff, and when you pressed the buttons zoom-voom-boom you got your answer and then everybody yelled with their deep throats woopee, yippee and maybe even sock-it-to-me."



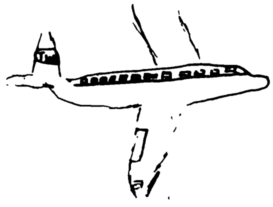
"Remembering eggacly what 'binary' means is something that is forever going to be on my mind."

"What a 'bit' is has a very short memory on my end."

Question: How long are people in school to learn about computers? Answer: "They could be anywhere from 5 feet and up."

"A bit of blarney is computer talk for all practical purposes invented by the Irish."

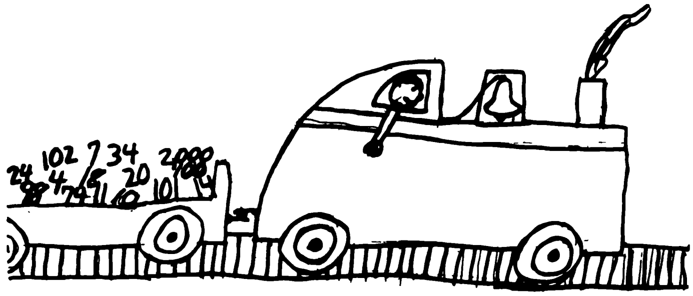
"I was so glad in my body to know that someday I would go to school to find out how to be a computer programmer. I had so many glad tickles in my stomach about it. Then with a sudden finding out that I also wanted to be a pilot, all my glad tickles went down my throat upside down and with a lump coming of sadness it was all over me being a computer programmer."



Question: How long have computers been in existence?
Answer: "Since the beginning of time and maybe even longer than this."

"The very first modern computer was built in the dark ages of 1930, in either the A.D. or V.D. times of history."

"If you like to fool around with figures alot then become a design engineer. My Uncle Henry is one, and he fools around alot with figures."



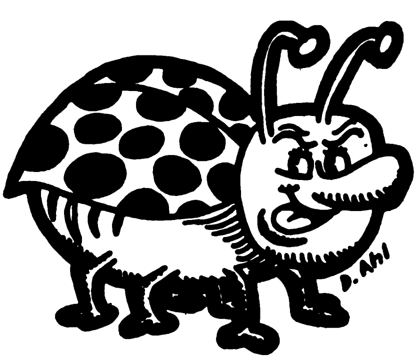
"A lady computer operator and a man computer operator are the same, only just the opposite in the you know where places."

"They are producing more and more people to work on computors anally."

"From now on, after learning all about computers, I'm going to think wonderful happy-that-you-made-it-so thoughts with a smile in my heart."

Aren't you?

Illustrations by Detta Ahl and Robert Green IV



Still a Few Bugs in the System

It bugs us here at *Creative Computing* when the mass media blame various problems on the computer. Even people in government, business, and schools find the computer a convenient scapegoat for problems actually caused by a programmer, keypuncher, faulty data collection techniques or other non-computer facets.

In this continuing column, we'll reprint articles or quotes which blame various catastrophies or problems on the computer. It's up to you, the reader, to decide whether the computer is actually to blame. Also, if you spot an appropriate item for the "Bugs" column, please sent it in.

Computer Fills Hotel With Angry Rumors

CHICAGO, Jan. 8 — Four thousand persons received letters yesterday thanking them for staying at the Oxford House, a downtown Chicago hotel.

Unfortunately, the 4000 letters had gone to the wrong addresses.

A computer error sent letters intended for Oxford House clients across the nation to 4000 Chicago residents. And in that friendly way computers have, each letter addressed the recipient by his or her first name.

"The phone hasn't stopped ringing all day," said Jerry Belanger, general manager of the hotel.

"One woman who received the letter is expecting her fourth child. Now she says her husband doesn't believe it's his," he moaned.

"Another woman who is suing her husband for divorce thought she might have some incriminating evidence to use. She was very upset when she found that the letter was a mistake.

"Some men called and demanded a retraction while their wives listened in on extension phones.

"The husbands were really the most irate. They got the letters but their wives opened them. Some couples said it was destroying their home life."

More than a few callers threatened lawsuits.

Belanger thought the mess had something to do with a mix-up in computer tapes by a letter-mail firm that had purchased address lists of department store credit card holders.

He said the computer was composing a letter of apology.

The Chicago Daily News

Unexpected Bonanza Thanks to Computer Error

ST. PAUL, Minnesota, Sept. 23 (AP) — It seemed like a bonanza to Joseph Pearson and his wife. The checks which came at intervals to his St. Paul home from the state totaled more than $25,000. Pearson says, "I saw 'Education Department' on the checks so I assumed they were from the Division of Vocational Rehabilitation. I didn't question them. I thought it was something I had coming."

The 52-year-old Pearson — who now manages several apartments — had injured his back on a construction job in 1969 and had been out of work for two years. He began taking a state rehabilitation program for job retraining. He dropped out of that in 1972. Three years later, the checks started coming.

The first check arrived in January 1975. Pearson thought it was a payment of benefits from the state because he couldn't work and had failed in the retraining program. Another check arrived in May of 1975. Then in September 1975, there was a check for more than $22,000. The fourth check arrived last May.

Pearson says, "I partied. I vacationed. I bought clothes and things that my wife and daughter didn't have during hard times."

Now Pearson is being sued by the state for the proceeds of the four checks that had been made out to him by mistake.

William Frietag, Superintendent of the Chandler, Minnesota school district, noticed a shortage of $25,585.76 in the state Transportation Aid account for his district. Pearson and the school district had been assigned the same computer number and the checks went to Pearson.

Michael Bradley, an Assistant Attorney General assigned to the state Department of Education, said it is the only instance in state history of such a computer error.

Computer Produces Shocking Invoice

So you think you have insurance bills? In Miami, Florida Baron Vladimir Kurt von Pousental received a shocking invoice for $5362. The 81-year-old motorist complained it was a bit steep for his 2-year-old chauffeured car. He also pointed out that his chauffeur had not collected 70 points for traffic violations as his insurance company claimed. The company replied that it was the computer's fault — one zero too many against the chauffeur.

Road and Track

Bankrupt by Computer, Frenchman Wins $300,000

GRENOBLE, France, July 17 (Reuters) — 'A fruit and vegetable wholesaler has been awarded $300,000 in damages after being driven to bankruptcy by a bank's computer error.

The computer of the state-owned Credit Lyonnais persistently rejected Eugene Rochette's checks to his suppliers on the ground of insufficient funds.

The suppliers protested to Mr. Rochette, who found his business, mainly with supermarkets, crumbling. Within weeks he was declared bankrupt.

Last year a lower court granted him $150,000 damages. The bank appealed, and an appeal court doubled the amount.

Woman Billed for Computer Goof

WASHINGTON — Though it wasn't her fault, a woman in New York owes the government $312 because a stupid computer put too much into her Social Security checks over a four-year period.

—*The Wall Street Journal*

Action Line

I received a notice from Social Security that my Survivor's Benefits were being terminated because I was no longer a fulltime student. I can't understand this because I'm registered at Henry Ford Community College for the fall semester so I should be eligible. I tried to find out what's going on but all I got was the runaround. Will you please look into this?—D.M., Dearborn Heights

Somehow computer convinced itself you were no longer continuing student, notified Social Security office personnel to send termination notice. Rejection meant you were broomed from benefit pool and Social Security stopped check flow effective August. Proper forms reinstating you are already in works and Social Security folks told Action Line you should receive $609 benefit check—covering August, September and October—in few weeks.

(From the *Detroit Free Press*. Thanks to Paul McCullough, Flint, Mich. for sending this in.)

Resources

COMPLEAT COMPUTER CATALOGUE

RESOURCES
AND
TOOLS

Reader Note

Please keep in mind that these entries are from the year 1977. All items, organizations, and miscellaneous may no longer be available or in existence. Check first with the addresses provided.

ORGANIZATIONS

JOIN FASST FAST

FASST (Forum for the Advancement of Students in Science and Technology) is a non-profit educational student organization representing individual members and chapters throughout the country and abroad. Dedicated to provide information for both the technical and non-technical student, an annual membership brings you: FASST NEWS, a quarterly tabloid covering a potpourri of science and technology developments and policy issues and FASST TRACKS, a quarterly newsletter containing organizational news, resource listings, opportunities for involvement, and listings of regional and national conferences. $5 year.

FASST Headquarters, 1785 Massachusetts Ave., N.W., Washington, DC 20036. (202) 483-2900.

NATIONAL CONSORTIUM FOR COMPUTER - BASED MUSICAL INSTRUCTION

The NCCBMI provides a Forum for the exchange of ideas among developers and users of computer-based systems for musical instruction, establishes and maintains a library of music courseware, reduces redundant effort among courseware and hardware developers, and provides consultation for new users of computer-based musical instruction. It meets semi-annually, and held its last meeting Feb. 22-25 at the Hotel duPont, Wilmington, Delaware.

Contact Fred T. Hofstetter, NCCBMI, Music Department, University of Delaware, Newark, Delaware 19711.

NASAGA

The basic goal of the North American Simulation and Gaming Association (NASAGA) is to advance an optimal, responsible application of the technique of simulation and gaming. The objectives of the Association are: to facilitate communications among persons interested in the field of simulation and gaming; to promote the training of specialists in the field of simulation and gaming; to facilitate communication between these specialists and policy-makers, students, and other concerned persons; and to promote the development of better techniques in the field of simulation and gaming.

Annual membership in NASAGA begins July 1 each year. Anyone becoming a member for 1977-78 at a cost of $12 will receive: a membership directory; a special NASAGA members rate at the 1977 NASAGA conference; a membership card; a one-year subscription to *Simulation/Gaming;* and an opportunity to contribute to expanding the use of gaming and simulation.

NASAGA % COMEX — Davidson Conference Center, University of Southern California — University Park, Los Angeles, CA 90007. (213) 741-6569.

CENTRAL STANDARDS LIBRARY

To help solve some of the standards problems in the hobbyist computer and microcomputer field, ALF Products is sponsoring a Central Standards Library as a means of standards information exchange for manufacturers, consumers, hobbyists, and others interested in standards. The Library will collect submitted standards and distribute them on a non-profit basis.

Manufacturers currently participating include: ALF Products, IMSAI Manufacturing, PolyMorphic Systems, Proko Electronics, Vector Graphic, and Video Terminal Technology. For more information on available standards, on how to submit standards, and on the library's services, send $1 (to cover printing and mailing costs) to: The Central Standards Library; c/o ALF Products Inc.; 128 S. Taft; Denver, CO 80228. You will receive a copy of the first CSL Newsletter and the first submitted standard (a parallel interface standard).

CONDUIT

CONDUIT, an organization which has been working to improve instructional computing at the college level since 1972, has been re-funded by the National Science Foundation to continue its activities for at least another two years. CONDUIT distributes quality instructional computing materials for higher education.

To perform this function, CONDUIT searches for quality computer-based instruction, peer-reviews such units to validate substantive content, tests software to ensure technical correctness and reliability, packages curriculum units so that both the software and teaching techniques are transferrable, and distributes these products on a national basis.

CONDUIT now distributes 41 curriculum units covering topics in Biology, Chemistry, Math, Management Sciences, Physics and Social Sciences. More units are being actively sought; the addition of approximately thirty packages per year is planned.

CONDUIT also developes guidelines and reports. One such product will be a second set of discipline-oriented "state of the art" reports, telling instructors how they may become involved in instructional computing and what is going on currently with computers in their own discipline.

For more information (including a complimentary copy of the CONDUIT newsletter, the Pipeline) write to CONDUIT, Box 388, Iowa City, Iowa 52240.

COMPUTER ARTS SOCIETY

The Computer Arts Society (CAS) was launched into existence with EVENT ONE at the Royal College of Art in 1969. As regards the Society's name, Alan Sutcliffe, Chairman of CAS since its inception, remarked in *Page 13*: "Some attitudes were agreed by those of us who first met . . . to talk about forming the Society. They were attitudes of neutrality and inclusiveness. That is why we chose the commonplace name of the society, even while agreeing that the term Computer Art was to be deprecated. It is still convenient shorthand for 'creative work in which a computer has been used.'"

In 1971, CAS established a policy of awarding honorary life memberships in the Society. John Whitney, Sr. (1971), Iannis Xenakis (1972), and Edward Ihnatowicz (1973) are the first three artists to be so honored.

The main communication of CAS is its newsletter, *PAGE*, which is published eight times each year. This newsletter contains commentary, reports, short articles, book reviews, and miscellaneous items related to the use of the computer in the arts. Most of the issues are published in England, but the membership is spread throughout the world.

U.S. and Canadian members of CAS also receive a single sheet bi-monthly announcement entitled *HAPPENINGS*; its purpose is to announce local shows, lectures and other events.

The Society aims to encourage the creative use of computers in the arts and allow the exchange of information in this area. Membership is open to all at £2 or $6 per year, students half price.

Alan Sutcliffe, 4 Binfield Road, Wokingham, Berkshire, England.

U.S. Branch (CASUS) Coordinator: Kurt Lauckner, Mathematics Department, Eastern Michigan University, Ypsilanti, Michigan 48197, U.S.A.

NATIONAL COMPUTER CLUB

Organized to provide communication among hobby-computer clubs, the National Computer Club has been formed. Over 45 club leaders and representatives met at the first Club Congress in June at NCC in Dallas. A committee was formed to discuss the services that should be provided by a national society for personal computing, and will meet at various computer shows.

The Committee's secretary is Jim White, 1202 River View Lane, Watertown, WI 53094.

MAGAZINES, JOURNALS, NEWSLETTERS

INTO EVERY LIFE A LITTLE RAIN MUST FALL

And if the rain is *Rain* magazine, consider yourself fortunate indeed. *Rain* is a lively publication which has for a subtitle "The Journal of Appropriate Technology." It runs about 24 pages per issue and carries mainly resource notes (similar to the Catalogue section of *Creative*) on a wide variety of subjects such as agriculture, architecture, education, energy, tools, health, soft technology, communications, food and nutrition, yes, and even computers. *Rain* started off with a bias toward the Pacific Northwest but today covers all of North America quite well. Annual subscription of 12 issues $10. "Living lightly" subscription rate is $5 ("living lightly" means you *really* can't afford $10). Sample issue $1.

RAIN, 2270 N.W. Irving, Portland, OR 97210. (503) 227-5110.

COMPUTERS IN CHEMICAL EDUCATION

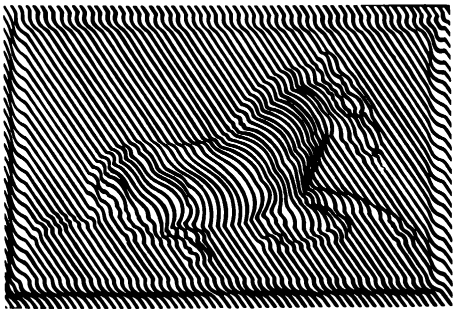
The Committee on the Role of Computers in Chemical Education of the Division of Chemical Education of the American Chemical Society (whew!) publishes a one page newsletter every two months. It contains brief (well they have to be with a name that big and a newsletter that short!) news items and plugs relating to CAI in chemistry. The newsletter is subsidized by the Division of Chemical Education of the ACS. Membership is $3.00.

R.W. Collins, Newsletter Editor, Eastern Michigan University, Ypsilanti, MI 48197

THE SPACE GAMER

A very professionally done magazine published quarterly. It deals with manually played space games and science fiction games (not computer games) and science fiction in general. A typical issue contains reviews of games, editorials, letters, and wantads. The Space Gamer is published by Metagaming Concepts which also makes science fiction and fantasy board games. $3 for six issues.

Metagaming Concepts, Box 15346, Austin, TX 78761



From. *Pattern Processing. A Further Rationalization of Sight* by Thorne Shipley, Leonardo, Vol. 8, 1, pp. 27-39, 1975

LEONARDO

Leonardo is a quarterly professional archival journal for artists, art teachers, and others interested in the contemporary visual or plastic fine arts particularly as related to science and technology. Illustrated articles by artists are carried covering virtually all techniques, content, and mediums. Even computers!

Leonardo also contains articles on developments in the other arts, on new materials and techniques, and on subects in aesthetics, architecture, education, the natural and social sciences, and technology. Extensive reviews section.

Subscriptions for institutions $55 per year, for individuals $15.

Pergamon Press, Maxwell House, Fairview Park, Elmsford, NY 10523.

The publisher writes that illustrated manuscripts are being sought on all aspects of computers in art. For information on submittal requirements, write Dr. Frank Malina, Editor, *Leonardo*, 17 rue Emile Dunois, 92100 Boulogne sur Seine, France.

SURE THINGS: DEATH, TAXES, AND LAWYERS

Are you associated with a data processing facility that keeps any kind of records on individuals? Are you developing or using proprietary software? Do you collect any kind of Federal, state, or local taxes? If you can answer "yes" to any of the above questions, then you probably should take a look at *Computer Law and Tax Report* edited by Robert Bigelow. This is a monthly 8-page newsletter that covers legal and tax matters related to data processing. It's expensive but it packs a wallop. Take a look at a sample copy before you subscribe. 1-Year subscription $48.00.

Warren, Gorham & Lamont, 210 South Street, Boston, MA 02111.

COMPUTER MUSIC JOURNAL

Devoted to the development of computer systems capable of producing high fidelity music, this new journal will cover such topics as: computer composition of music; design of real time playing instruments; real time input controllers such as Keyboards, joysticks, and new controllers, homebrew computer music instruments.

The first issue of the journal will be about 50 pages in length, and will increase in size and scope as interest dictates. The first issue should be out by now. Bimonthly $14.

Computer Music Journal, PCC, Box 310, Menlo Park, CA 94025.

CRYPTOLOGIA JOURNAL

CRYPTOLOGIA, a journal devoted to all aspects of cryptology, will be published four times a year and will contain research papers, survey articles, personal accounts, reviews, educational notes, and problems. Some of the areas which will be discussed are mathematical, computational, literary, historical, political, military, mechanical and archeological aspects of cryptology. Editors are: Cipher A. Deavours, Department of Mathematics, Kean College; David Kahn, Department of Journalism, New York University; and Brian J. Winkel, Department of Mathematics, Albion College. Quarterly, $16/year.

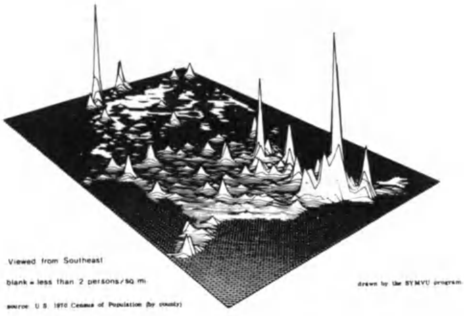
CRYPTOLOGIA, Albion College, Albion, Michigan 49224.

UNEARTHLY WONDERS

UNEARTH is a new science fiction magazine devoted to publishing stories from new writers. It runs the entire gamut — speculations, fantasy, horror, computers, robots, etc. *UNEARTH* also includes book and movie reviews, a science column by Hal Clement, and a special feature from contributing editor Harlan Ellison (in issue #1 this was Harlan's very first story, "Glowworm"). *UNEARTH* is also seeking stories and artwork. Published quarterly, $3.50 per year. Sample copy $1.00.

UNEARTH, Suite 190, 102 Charles St., Boston, MA 02114.

BOOKS AND BOOKLETS


Viewed from Southeast
blank = less than 2 persons/sq mi
source: U.S. 1970 Census of Population (by county)
drawn by the SYMVU program

LOST? DRAW A MAP

The Laboratory for Computer Graphics and Spatial Analysis within the Graduate School of Design at Harvard University has just released a new edition of LAB-LOG, its catalog of computer programs, data bases and publications. Research at the Laboratory is principally concerned with the analysis and graphic display of geographic data used in the planning process. LAB-LOG describes various products which have resulted from this work and which are currently available for distribution to universities, government agencies and private organizations.

LAB-LOG includes a description of six different computer programs for use in the graphical display of spatial data via a line printer, line plotter and cathode ray tubes. The programs are written for IBM 370 series computers and range in cost from $575 to $1200. A wide variety of cartographic (x-y coordinate) data bases are also described. Publications are available on the subjects of automated cartography, theoretical cartography and theoretical geography. Publications cost from $1.00 to $12.00. LAB-LOG also contains a brief description of the Laboratory's history, research directions and operating policies. Copies of LAB-LOG $1.00 prepaid.

The Laboratory for Computer Graphics and Spatial Analysis, 520 Gund Hall - Harvard University, 48 Quincy Street, Cambridge, MA 02138.

BIBLIOGRAPHY ON COMPUTER IMPACT

An annotated bibliography on Computer Impact on Society is available on microfiche from the Virginia Institute of Marine Science. The bibliography contains more than 2,000 entries of books, articles and other items. Most items are annotated. The listing is alphabetical by author; however, the entire system is stored in a hierarchical storage and retrieval system at the University of Wisconsin and special-purpose subsets of the collection may be obtained. The complete set is $2.30 for individuals in the United States, and $3.30 for others.

Gerald L. Engel, Dept. of Computing and Statistics, Virginia Institute of Marine Science, Gloucester Point, VA 23062.

COMPUTER-ORIENTED BOOK BIBLIOGRAPHY

More than 225 new books are listed in the tenth edition of the *Annual Bibliography of Computer-Oriented Books*, published by the University of Colorado. Even with all books before 1971 deleted, the bibliography still contains more than 1,000 books from 210 publishers.

Books are separated into 55 categories and catalog according to type and style of presentation. A new category was added this year: program design. The bibliography is $4; or $5 if an invoice is required.

Computing Newsletter, Box 7345, Colorado Springs, CO 80933.

EDUCATION MATERIALS CATALOG

Catalog contains 9 sections describing various services, books, and products of NWREL. The computer section describes a computer careers book, teacher's guide, and student guide; two REACT courses and booklets including the *Teach Yourself Basic* Booklets (Relevant Educational Applications of Computer Technology; PLANIT (A CAI author language); and several pending new products.

NWREL is a non-profit organization involved with many disciplines (adult education, bilingual education, communication, reading, language, research, etc.), over 800 member institutions, and a wide variety of sponsors. Prices on NWREL products are very reasonable. Catalog free.

Northwest Regional Education Laboratory, Dept. J, 710 S.W Second Ave., Portland, OR 97204.

CATALOG OF THE FUTURE

A huge 40-page tabloid catalog of books, magazines, learning materials, games, and cassettes is now available from the World Future Society. Catalog lists books by major and not-so-major publishers. There's something for everyone here — topics include computers, business/industry, communications, education, environment, forecasting, simulation/gaming, systems design, and many more. A worthwhile catalog and the price is right — free.

World Future Society, 4916 St. Elmo Ave., Washington, DC 20014.

COMPUTER-RELATED CRIME

Six new reports on computer abuse, system vulnerabilities, applicability of Federal and state penal laws, proprietary rights in software, and privacy law effects are available from the SRI Project on Computer Abuse headed by Donn B. Parker. Announcement of reports available and order form are available from Allison Brandt, Report Production, Bldg 300 B5, SRI.

Donn Parker is also interested in information about specific computer-related crimes and other abuse (vandalism, property theft or fraud, financial theft or fraud, unauthorized use of services, etc.). Write him at SRI or call (415) 326-6200 X2378.

Stanford Research Institute, 333 Ravenswood Ave., Menlo Park, CA 94025.

PERIODICAL GUIDE

The *Periodical Guide for Computerists 1976* is a 20-page index to articles from 15 magazines read by computer hobbyists. The magazine include *Creative Computing, Byte, Dr. Dobbs Journal, SSCS Interface, Interface Age, Popular Electronics, Radio Electronics, PCC, 73, Electronics, EDN, Electronic Design, Digital Design*, and the ill-fated *Microtrek*. Over 100 subject headings are used with more than 1,000 articles, book reviews, letters and editorials. $2.50.

E. Berg Publications, 1360 W.S. 199 Ct., Aloha, OR 97005.

ACM ADMINISTRATIVE DIRECTORY

The 1977 edition of the ACM Administrative Directory of Chairman of University and College Computer Science Departments and Directors of Computer Centers provides names, addresses, and telephone numbers, and lists computer-science and data-processing degrees offered and major on-site computing equipment. The officers and key staff members of related computer organizations are included. The directory consists of more than 2300 names, and is $7 for ACM members, $9 for others.

ACM Order Department, P.O. Box 12105, Church Street Station, New York, NY 10249.

UNDERGROUND BUYING GUIDE

The *Underground Buying Guide for Hams, CBers, Experimenters and Computer Hobbists* lists over 600 sources that cater to the electronics hobbyist. The first section contains an alphabetical listing of firms, capsule descriptions of their offerings, prices, complete addresses and phone numbers.

The second section contains 200 categories of parts, services, etc., with a cross-reference to the first section. The third section is a geographical cross-reference.

We hope most of the listings are more accurate than those for *Creative Computing* and *Byte*. The *Creative* listing in Section 1 mentions only books (and none of our own) and ignores the magazine. *Byte's* listing in Section 1 is OK, but in Section 2 *Byte* is cross-referenced under books, not magazines; *Creative* is not cross-referenced at all in Section 2. We checked about two dozen hardware manufacturer listings and found them generally accurate. The biggest problem is omissions in a fast-moving field; obvious ones we spotted included TDL, ECL, North Star, and Ximedia.

Despite the Criticism, there's nothing else like it and you may find it a worthwhile investment. $5.95 postpaid from PMS Publishing Co., 20440 Town Center Lane, Cupertino, Cupertino, CA 95014. Or order direct from *Creative Computing*. Same price.

DESIGN YOUR OWN GAME

The second edition of *Design Your Own Game* has been published by Harvard W. McLean and Michael J. Raymond, two Ohio educators. This 94-page soft-cover book goes into both simulation and instructional games, and has chapters on selecting formats, objectives, limiting factors and interaction sequences in the model, rules, evaluation, writing a manual for the game, etc. The existing games of each type are presented and modified. The last pages provide helpful directories of simulations and games, organizations, periodicals, and companies and catalogs. $3.95.

The Simulation and Gaming Association, 4833 Greentree Road, Lebanon, OH 45036.

PROGRAM ABSTRACT DATA BASE

The National Computer Program Abstract Service (NCPAS) has 25,000 abstracts in its data base, said to be the largest retrievable data base of its type in the country. These abstracts are on computer simulation models, application and computational programs, and information-retrieval systems covering "all fields of knowledge from business, government, industry, military, and universities."

The computer-program information is disseminated in two forms: as a quarterly program-index newsletter at $10 a year, and as special abstract reports on each subject area at $19 for up to the first 200 abstracts and $6 for up to each additional 200 abstracts.

For a free report index, write to NCPAS, P.O. Box 3873, Washington, DC 20007.

DESIGNING WITH MICROPROCESSORS

This tutorial deals with the principles and practises of microcomputer design, covering such topics as chip architecture, microprocessor selection criteria, software aids, development systems, microprocessor applications, networks, busing strategies, and distributed intelligence. To IEEE members, $7.50; nomembers, $10.

IEEE Computer Society, 5855 Naples Plaza, Suite 301, Long Beach, CA 90803.

ATSU

The Association of Time-Sharing Users is a professional organization whose purposes are to supply current information to time-sharing users, to provide information about the various products and services offered by remote computing suppliers. to provide a forum for discussion of topics pertaining to remote computing and interactive time-sharing. It provides four publications: a newsletter, a Press Review, "Interactive Computing," and "Interactive Computing Directories." Membership $20.

Hillel Segal, President, ATSU, 75 Manhattan Drive, Boulder, Colorado 80302

MISCELLANEOUS

LASER MUSIC SPECTACLE

A Soleil Laser Music Spectacle is made up of three types of laser movement. For the pre-programmed type, every movement of the laser light is planned, and a large-scale digital computer is used to compute the millions of commands required for a Soleil show. The program is stored on four-channel audio tape, with two channels for stereo music, and two for control information for the lasers. The laser-light movements can also be controlled directly by an operator; these movements can also be combined with a pre-programmed tape.

The Soleil Laser Music Spectacle is available for engagements. Contact Kim Rogers, RR #2, Box 103, Bloomington, IN 47401. (812) 336-8222.

COMPUTER PHILATELY

A group of stamp collectors have put together a list of stamps relating to computers. The list contains more than 130 stamps. Free, but include self-addressed stamped envelope with 24¢ postage on it.

Robert V. Boos, 66 Crescent St., Hicksville, NY 11801.

CALCULATORS

COMPUTERIZED MATH DRILL

Part calculator and part computer, the Classmate 88 from Monroe uses an individualizes instruction approach to enable students to generate ulimited drill and practice routines in over 70 computational skills. Flowchart instructions allow students to work without supervision. All drill and practice programs are hardwired into the machine, with special keys to select the subject area. Automatic scoring permits the teacher to evaluate student progress.

Monroe, The American Road, Morris Plains, NJ 07950.

7 8 9 +
4 5 6 x
1 2 3 –
SET 0 GO +

LIL' PROFESSOR FOR LIL' KIDS

The TI "Little Professor," a calculator-based learning aid, contains more than 16,000 pre-programmed basic math problems for children ages 5-9.

"Little Professor" problems can be selected according to degree of difficulty by the user. Once the type of problem and degree of difficulty is selected, the machine presents the problem (with function and equal sign) in the large VLED display. The user keys in an answer. If correct, the complete equation appears for one second, and the next problem in sequence is displayed. If incorrect, the problem reappears.

If a wrong answer is entered three times, the "Little Professor" automatically displays the correct answer. The answer remains in the display until the user presses the "GO" key. If the next problem is also answered incorrectly three times, the correct answer would again be displayed.

The "Little Professor" will, after ten problems, display a score (number out of ten correct). This "score" display would be followed by the next set of problems in sequence.

The "Little Professor" has addition, subtraction, multiplication and division sequences in variable degrees of difficulty. A user works at his own speed, and considers the machine a game, as well as a learning aid.

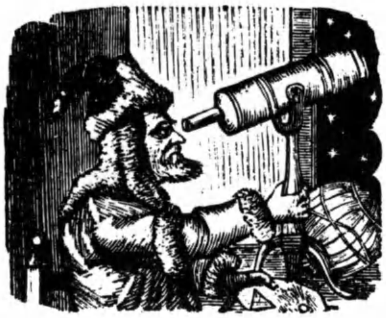
Texas Instruments (Little Professor Inquiries), P. O. Box 5012, M/S84, Dallas, Tx 75222. Nelson Brooks (214) 238-4203.

MATCHBOOK SIZE CALCULATOR

The Micro-Mini is claimed to be the world's smallest electronic calculator. The unit measures 1.675 x 2.375 x 0.5 inches — about the size of a matchbook. The calculator weighs 1.2 oz, including the silver-oxide battery. The unit can add, subtract, multiply and divide and has a constant for multiplication and division. It has an eight-digit liquid-crystal display. $29.95.

Casio, Inc., 15 Gardner Road, Fairfield, NJ 07006. (201) 575-7400.

COMPENDIUM



Spinrad's Galaxy

A Berkeley astronomer named Hyron Spinrad recently identified a super-giant eliptical galaxy which he predicts is at least half the age of the universe. It is, claims Spinrad, the most distant stellar object of its kind yet found.

Known for the last 20 years as an intense radio source in the autumn sky north of the Pleiades, the galaxy, designated as 3C123 on star maps, has finally appeared as a dim spot on a computer-processed photograph. The cluster contains thousands of billions of stars and perhaps three sister galaxies whose light has been traveling through space for *eight billion years*—about half the estimated age of the universe.

Prof. Spinrad discovered the object by utilizing a new computerized "image tube scanner" system, which collects spectroscopic data over several nights of observation and then subtracts the "glare" (actually brighter than the image itself) effects of background light. The scanner system is attached to the 120-inch telescope at the University of California's Lick Observatory.

Computer Projects 21st Century Hunger Chaos

CLEVELAND—(UPI)—A computer survival program at Case-Western Reserve University, capable of plotting for many years in advance the economic fortunes of nations, forecasts world chaos in less than a century because of food shortages.

Six private foundations have provided $790,000 in funding for the university computer center to develop computerized programs that would help countries avoid economic disaster.

Prof. Mihajlo Mesarovic said the school is bidding to become the world center for computer planned survival of some of the major nations bordering the Pacific Ocean.

Graphics Display System

Combining the features of an automatic draftsman and a versatile computer, with unlimited capabilities for enhancement of the human imagination, is the Spar/Graphics Display System, from Scientific Process & Research, Inc.

This system, for example, can be of assistance to designers of artwork, logos, and layouts, utilizing computer graphics, interacting to offer selective choices in great variety. Small patterns can be created, then displayed in any combination to determine overall effect. Thousands of *any* combinations can be tested in the time it takes to draw a single image or pattern. Complex patterns or concepts can be self-generated by the system to expand the user's own imagination. Complex objects and shapes can be created, altered, copied, repeated, enlarged or multiplied.

Points, lines, curves, unusual or complex shapes are displayed on the screen. A single command duplicates the figure, if desired. Another will move it or enlarge it, reduce it, or rotate it. Three-dimensional effects can be added, as well as new features, at any stage. Designs can be named and recalled easily. Moreover, several figures or concepts can be combined and given a new name. One can display hundreds of designs, drawings, or graphs, in minutes, and select the one wanted, then print it. Draw one element, and in seconds, a whole pattern is generated. Elements can be altered, rotating some, enlarging other. Forms or figures can be stored, even entire patterns or layouts, by a single command on a magnetic disk included with this system, which is from Scientific Process & Research, Inc., 24 North Third Ave., Highland Park, NJ 08904.

Computers in Architecture

The time is coming, and it may be in the very near future, says Charles Eastman, when the computer will become as common an architectural design tool as the T-square and triangle.

Eastman, a professor of architecture, computer science and urban affairs at CMU, can sit down at a console and by giving the right commands, call up various drawings of a building to a computer driven TV screen and make changes in the building's configuration.

It only takes a minute or two, for instance, for Eastman to call up on the screen, any plan, section, evaluation or perspective of the University's administration building, a six-story office complex. The computer can provide perspective or orthographic displays of the structural elements, exterior panels, plumbing, mechanical equipment or interiors, for example. Any number of elements such as the heating system ductwork can be added or subtracted by pressing a few keys.

Eastman's system would not do away with the traditional architect's function of design; he would still have to use his technical and aesthetic judgment. "A great part of the architect's cost now is in the time it takes to produce the drawings," says Eastman. "If we can get a computer to produce those detailed drawings, it will cut down drastically on the time and, therefore, the cost of an architect's work."

Eastman says housing developers might find they can offer custom built homes instead of houses that all look alike. "The possibilities are that you could sit down at a computer terminal and work out a custom designed house with drawings, parts lists and possibly even construction schedules prepared automatically."

—Carnegie-Mellon Alumni News

Cal Students on Leave Program to Work at IBM

Selected computer-science students at the University of California at Berkely have been taking half-year leaves of absence from college to gain work experience with computers at IBM's San Francisco Data Center. Called the Co-op Program, the work program is designed to give the students an opportunity to apply their classroom knowledge to meaningful and real jobs with IBM's Data Processing division.

"The students work on real problems and provide a valuable assist to our regular staff," said Earl Ness, manager of IBM's San Francisco Data Center. "At the same time, the students gain solid work experience and earn pay to help with their college expenses."

Whither Computers?

Despite heavy investment and numerous successful experiments with computerized instruction techniques in school systems nationwide, "computers have failed to make more than a minimal impact on education"—primarily because of financial constraints, according to Robert G. Scanlon, executive director of Research for Better Schools, Inc.

This was the apparent consensus of computer companies assembled recently in Washington, D.C. for the First International Learning Technology Conference. After a decade of trying to get into the public education business, most companies have decided to temporarily forego public education and concentrate instead on business.

In an article in *The Washington Star* (July 23, 1976), Scanlon was quoted as saying that in the 1960s the federal government invested nearly $100 million in experiments to promote electronic teaching devices. While teachers and administrators endorsed use of computers, "when the funds ran out, school after school returned to the old way of doing things."

Computer companies now hope that as more and more businesses install computers to train employees, public schools will eventually see electronic technology as an economical approach to teaching.

—The Science Teacher

Computer Game Helps Students Understand Congress

Out-guessing the US Congress with the help of a computer can be fun and educational too. Designed for secondary- and college-level students, "The Congressional Game" was developed by a Univ. of Pennsylvania professor, Dr. Robert Zemsky, and the Uni-Coll Corp., a Philadelphia computer utility.

With the new program, the history student can simulate what it was like to be sitting in Congress a century ago. At that time, Grant was President and the Congress was debating the Korean seizure of an American naval vessel, civil rights, import tariffs and federal subsidies for railroads. The game was developed with two purposes in mind— to help students understand how Congress works and to help them understand how historians work.

The program presents the student with various authentic historical facts and constantly engages the student in the selection, evaluation and use of the information to develop a hypothesis regarding the outcome of a roll-call vote. The winner is the student who has the most success in refining the hypotheses about how the House will act when seen at a realistic, daily level.

Brain Monitor

The psychology department at De Anza Community College in Cupertino, California, uses a Hewlett-Packard RTE-II minicomputer system to teach classes in data processing and statistics, and to help conduct laboratory-based psychological experitory-based psychological experiments, including human and animal learning and cognition. The RTE-II system, through an analog-to-digital converter, receives information from a number of physiological and biofeedback testing devices.

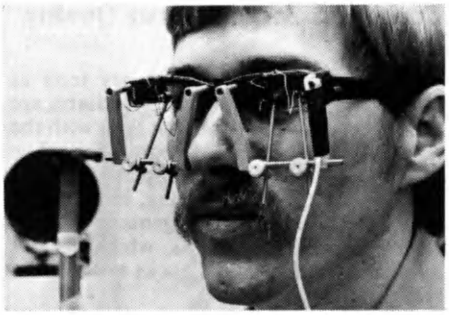
In experiments with humans, the RTE-II system is involved in biofeed-back. Through interfaces, it collects information on, correlates and plots the heart rate, galvanic skin response, EEG readings and temperature of each subject. In the photo, Frank C. Savage, professor of psychology, conducts computer-based measurements of the brain-wave activity of Thomas D. Carrell, senior psychology technician.

Computer Check at the Border

As automobiles stop at the customs checkpoint in the sleepy village of Roma, Texas, few drivers realize that their license-plate numbers have been keyed into a U.S. Treasury Department computer 1,200 miles away in San Diego.

Within seconds, customs inspectors know if the vehicles have been stolen and whether their owners are suspected of involvement in a crime just about anywhere in the world.

Similar facilities are located in other cities along the Mexican border, which are also linked by a voice circuit.



Eyes Tracked by Computer For Hints of Disease

A new eye-movement monitoring device, developed by a Carnegie-Mellon University biomedical engineering professor and three physicians at Pittsburgh's Eye and Ear Hospital, may eventually detect crippling diseases in their early stages.

The Optical Data Digitizer, or the "ODD Tracker," is a camera interfaced with a minicomputer which scans across the eye and notes the position and vertical/horizontal motion of the eye. "It's biggest advantage is that you don't have to touch the patient," says Dr. Bahill.

Watching the eyes is an increasingly important way of determing the health of the patient. "The eyes are the windows to the brain," says Dr. Bahill, who first became interested in eye movements while a graduate student at the University of California at Berkeley.

Dr. Bahill contends that the brain and eye work on a "time optional system" where the brain focuses the eye on an object and shifts it to another object in the least amount of time. He believes that persons suffering from diseases develop abnormal eye movement patterns. These patterns are often the first symptoms of such diseas as Multiple Sclerosis, a degenerative disease of the central nervous system; Parkinson's Disease and Huntington's Chorea, two muscular disorders accompanied by tremors, and Duane's Syndrome, a form of eye paralysis. For example, the ODD Tracker shows persons suffering from Multiple Sclerosis will fall short of a target with one eye and over-shoot the same target with the other eye.

Computer Controls Drug Quality

Drug quality and consistency tests at Miles Laboratories in Elkhart, Indiana, are being performed for the first time with the help of a small IBM computer. The System/7 enables quality-control chemists to do more extensive testing, in a shorter time, on clinical test reagents and on a variety of Miles products, which include Alka-Seltzer, Bactine, Chocks and One-A-Day vitamins.

The System/7 is linked to instruments such as auto analyzers, gas chromatographs and amino-acid analyzers to test for such things as stability and consistency of product ingredients.

Before the system was installed, instrument readings were recorded and analysed manually, which took half a day for testing and the rest of the day for comparative analysis. Now the tests are performed throughout the day and results are known minutes later. At the same time, the system eliminates costly transcribing errors that plague manual testing operations.

Outsmarting Computers is Profitable Hobby for Floridian

WINTER PARK, Fla—Tired of stamp collecting? Try outsmarting computers for a hobby, the way Christopher Cossette does.

A computer hobbyist with a different idea, Cossette proves the fallibility of computer systems - any computer system. And so far he seems very good at his hobby.

Cossette claims to be able "to get in anybody's program with a few telephone calls." Such companies as Exxon, Master Charge and Bank of America all have the same inaccuracies built into their systems, he said, although he refused to disclose any further details.

In 1974, Cossette obtained $110,881 from a Canadian department store over several months' time. It took him a while to convince the company what was happening and, in the meantime, he earned enough interest from the money to pay for a sports car and a swimming pool.

By working at his avocation, Cossette also managed to acquire three credit cards against which nothing is ever billed, he said. Airlines have credited him for tickets he never bought and stores have sent him duplicates or triplicates of items without charge.

Cossette, who claims to have done nothing illegal, does not think highly of the computer industry and accused it of "gross negligence."

"Computers are no smarter than the people who program them, and that quality must have fallen off recently," he said.

"I think the industry has improved on the Peter Principle," he added. The Peter Principle is the concept which states that workers rise to the level of their incompetence.

For those panting to take up the hobby, Cossette is publishing a book on the subject, which will disclose how he managed his success.

—Computerworld

Sewing Machine Uses Microcircuit

A custom-designed microelectronic circuit has made possible a home sewing machine that has more sophisticated sewing features, is easier to use, and contains 350 fewer mechanical parts than conventional models. The microcircuit permits the Athena 2000 sewing machine by Singer to be programmed with control buttons to sew basic practical and complex decorative stitch patterns, repeat a single pattern unit and stop, and do buttonholes completely automatically. Circuitry on the chip coordinates needle and fabric movement for optimum width, length, and density of 24 stitch types. Safety logic circuits help prevent damage to the needle and the machine.

The LSI/MOS chip contains 2,000 transistors and was produced by American Microsystems. The chip is primarily a ROM that permanently stores the instructions for sewing the 24 different patterns.

According to Jack Wurst, Singer manager of electronic development, "To duplicate the Athena 2000 machine using mechanical systems instead of electronics would be practically impossible. It could be done; it would be an extremely large, cumbersome, and expensive machine."

Computer Helps Feed Cows

Cows on a Dutch model farm are reported to be healthy and happy after a year's trial of a computer-based feeding system developed by the DACA Electronic Engineering and Contracting Company of Leystad. The system was developed to solve the feeding problems of large dairy farms, where some cows need more feed supplement than others. A transmitter attached to the cow's collar sends a signal, via a receiver in the cow's manger, to a computer that checks stored data and decides whether she needs any supplement at the time. If Daisy's milk production has been low she receives a predetermined quantity of supplement.

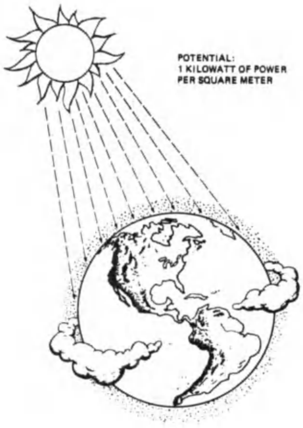
Computer Helps Archeologists Unearth Pre-Historic Communities

Computer technology has joined the pick, shovel and drill to help Northwestern University archeologists at the largest excavation site in North America. Archeologists working at the Koster site in southern Illinois rely on a Control Data 6400 computer 300 miles away at Northwestern University to keep track of their findings and even determine where to begin the next excavation.

In addition to information unearthed at the Koster site, pertinent data recovered from some 800 different archeological sites in the 2,800-square-mile research area is entered through a terminal for relay to the CDC computer. At Northwestern's Vogelback Computing Center, the 6400 uses a university-developed data-base program known as RIQS (Remote Information Query System), designed to handle the varied data of individual researchers.

From information entered on terminals at the excavation headquarters, the CDC computer built a file for each of the 800 sites. Each file is structured to hold 145 items of information about the site and the excavation results. Site description includes name, location and size of the excavation, names of the archeologists involved, where the artifacts from the site are stored, and what has been photographed.

The survey information sent to the computer relates to soil conditions, evidence of cobbles and limestone (indicates cooking, pottery and tool making) and of animal bone (means there is favorable preservation at the site). Of the 145 slots of information storage for each site, 123 are for listing the artifacts uncovered—the data most important to archeological analyses.



IBM Scientists Apply Computers to Harvesting Solar Energy

With a limitless and free source of energy staring us in the face, why is humanity relegated to digging holes in the earth in an effort to find enought fuel to power society? Spurred by shortages of fossil fuels and an ever-increasing need for more energy, scientists are now using the most sophisticated tools available to answer this question-to explore the obstacles that have thus far prevented mankind from efficiently harvesting solar energy for widespread use.

At IBM Corporation's Palo Alto Scientific Center, research specialists have initiated solar-energy studies by applying powerful computers to the problems of tapping the sun as a widespread and economical source of energy.

Between 10 and 50 percent of the light reaching the earth's surface is indirect, diffused sky light—depending upon atmospheric conditions—rather than direct sunlight. "We know something about the characteristics of diffused sky light energy, but not enough for efficient harvesting," said IBM's Dr. J.V. Dave, a radiation specialist. "Scientists hope to develop ways to harvest diffused sky light in conjunction with direct sunlight. But before that can occur, it is essential that we learn more about the nature of sky light energy."

At best only about 10-12 percent of the solar energy reaching a given spot on the earth can be harvested. That's because the sun's energy is spread over different wave lengths and no solar cell can capture all forms--from infrared or heat energy to near ultraviolet energy. To manufacture cells that capture even 20 percent of the available solar energy becomes prohibitively costly.

Because of their ability to stimulate reality and to solve gigantic and complex mathematical problems, computers are viewed by Dr. Dave and other scientists as valuable aids for learning more about solar energy. Dr. Dave hopes to simulate characteristics of sky light energy under representative atmospheric conditions and study their impact on solar cells. The results of such experiments and simulation studies could then be used by others who are exploring various alternatives to be considered in advance of designing and building actual solar-energy plants.

Computer-Error Merchandise

The *Saturday Review* classified ads now include a section on "Computer-Error Merchandise" which has offered, among other things: several hundred bicycles with handlebars fore and aft; a similar number of plastic shampoo bottles mistakenly filled with vichyssoise; and 38,000 copies of "Mein Kampf" in Polish, minus vowels.

MEDPA Newsletter

Mind-Reading Machine

The government is developing mind-reading machines that can show, among other things, whether a person is fatigued, puzzled or daydreaming. If the project lives up to its promises, the machines could be installed in airplane cockpits in a few years, to warn the pilot that his mind is wandering and he is not performing essential duties.

Since 1973, the Advanced Research Projects Agency at the Pentagon has been studying ways to plug a computer into a person's brainwaves. So far, scientists have been able to determine a person's alertness, and how he perceives colors and shapes. Current research is on the use of brainwaves to control machines.

Other research is aimed at using brainwaves to discover how a student learns and when he is most likely to learn, and also to improve computer-based teaching programs.

So What's the Bad Word?

English suffers endless indignities, but it does not suffer in silence. Some 12 years ago, the publishers of the American Heritage Dictionary, seeking protection in numbers, collected together a group of 150 authors, critics, editors, historians, etc. and dubbed them the "Usage Panel." The Feb. 24 *New York Times* reported some of their comments to the latest atrocities, two of which are reprinted below.

Is it acceptable to use "free up," as in "a new copying machine that will free up your secretary"?

J.K. Galbraith, economist: "Indecent, even obscene."

Jacques Barzun: "She's a loose woman already."

Nat Hentoff, author: "I think there's a chance of nipping this one. It should be shunned up."

"Input" is used as equivalent to "data or information" in charting a course, as in "The President had access to varied input," and to "an active role" as in "The nominee declared that he had no input in adoption of the plank." Yes or No?

Jacques Barzun: "...jargon—and very vague, since input can mean anything from a Congressional appropriation to a frankfurter at lunch."

Nat Hentoff: "...mechanical shorthand that rusts thought."

Bruce Catton, historian: "...particularly offensive form of the social science jargon."

John Fowles, author: "A Watergatism (mechanistic barbarism)!"

Lewis Mumford, author: " 'Input' has a legitimate use in computerdom— otherwise it should be shunned. It is the equivalent of 'y'know' for those who don't know the right word."

Berton Roueche, author: "I accept 'output,' but—I don't know why— 'input' turns my stomach. Maybe it's the people who use such words."

Pierre Berton, author: "I do not mingle socially with people who talk this way and would not expect my readers to stick with me if I used it."

Reuven Frank, television producer, voting no: "If there is output there must be input. If there is outcry there must be incry. If there is outlaw there must be inlaw. So the reasoning is junk."

Gilbert Highet: "...carries an objectionable image of a politician as a sort of I.B.M. machine passively receiving whatever people stuff into his slots."

Peter De Vries: "...the thought of putting information into a President is a little grotesque."

Red Smith: "This usage brings a violent output of nausea here. Couldn't the President have access to advice instead?"

Computer Debut at Carnegie Hall

A new opera recently opened in Carnegie Hall, and its soprano soloist has won modest acclaim for its debut. The up-and-coming new soprano is a computer, and it starred in an opera titled "Mar-ri-ia-a", a miniature opera for soprano computer and chamber ensemble.

The piece was written by Joseph Olive who also has experience in electronically synthesizing human speech. It is a science fiction comic opera, and its plot deals with a woman scientist who falls in love with the computer that she has built. The computer, unfortunately, is both crazy and a sex maniac; so, the heroine must destroy her only love in order to escape an unspeakable fate.

A *New York Times* reviewer said that "Musically, the piece is rather thin and could use more definite shape and mood in the last few minutes"; however, he did not fault the performance of the computer.

Educational Net in SW Germany

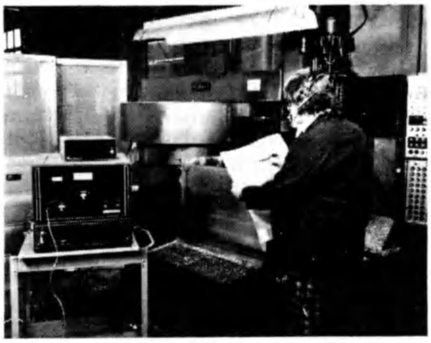
A Cyber 174 computer system will be installed late this year at the University of Stuttgart in Southwest Germany. However, this is not your ordinary system; initially, it will include forty interactive terminals installed in classrooms, laboratories, and service centers as well as smaller universities and research centers in the southwest region.

The system is valued at $4.5 million and is manufactured by Control Data Corporation. It will provide educational data processing services in Southwest Germany. A CDC 6600 has been in use there since 1968, and the new system will be linked to it to form a multi-mainframe operation sharing common peripherals.

The university computer center will serve 10,000 students' academic needs and will support a high level of research activity in areas such as nuclear power plant design, theoretical chemistry, and structural analysis.

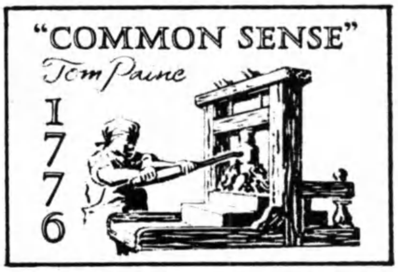
"Drill a ½" Hole in the Right Corner"

Hold on to your hats. It is now possible to program automatic machines, at least highly structured numerical control machines, by speaking commands to them. After developing a supermarket checkout system that can understand any clerk's voice pattern, Threshold Technology Inc., Cinnaminson, NJ, has come up with a technique that enables factory personnel with little or no programming experience to quickly prepare a fully verified, punched paper tape program for automatic machine tools. The programmer simply speaks each programming command in sequence into the microphone, using normal English words. The VNC–100 then automatically "decodes" the information into a machine-compatible format. It is estimated that this technique could cut programming time and costs by 30–80%.



An electronic display in the system instantly flashes each command given for a positive verification or correction and even displays the next entry required. The system, complete with software, a standard postprocessor package, operator and programming manuals, is priced at $20,300 for single units.

A similar voice-operated computer system is being used by EMI Ltd. of London to capture monthly financial information. A user repeats a new word 5–10 times into a noise-cancelling microphone, according to Threshold Technology.



The Plight of Strathclyde

In Spring 1975 after many disheartening delays, 10 primary schools in Glasgow with some 1800 students initiated an extensive CAI program similar to the very successful program in Chicago elementary schools (see photo). But unfortunately for Univac and Computer Curriculum Corp., suppliers of the hardware and software respectively, and for Glasgow by July 1975, a few short weeks after the project started, economic conditions in Scotland had deteriorated so drastically, that the system was ordered removed.

Even in the short time the system was installed, it impressed most who saw it in operation, including several cynical city counsellors who had initially opposed it. Nevertheless, the counsellors felt they had no option but to abandon the system as its £200,000 per year cost could just not be afforded.

A Scottish newspaper had this to say about the Strathclyde Region (of which Glasgow is the center). "Its malaise, social deprivation and its concomitant educational retardation, must be tackled and conquered if the region is ever to attain parity with the rest of Britain, far less prosper, so in this respect the decision was not only regretful but ominous. It is a situation not unlike a warship in battle jettisoning its guns and ammunition to prevent its sinking. The economy of Strathclyde will not improve while its education flounders — one cannot be divorced from the other."

ASSISTED
READING
MATH

Buy-Heinz-Pickles-Or-I-Will-Crush-You.

Sales promotion robots are being offered by Quasar Industries (380 Main Street, Hackensack, N. J. 07601). The androids are 5'4", weigh 240 pounds, are mobile, programmed with a distinctive humanoid voice, and have a pulsating light in sync with the voice.

Computers vs. the Postman

Computers used in evaluating routes of letter carriers could be the cause of the next Postal Service strike. In Kokomo, Indiana and Portland, Oregon, men have followed the routes of the towns' letter carriers, measuring distance and the lay of the land. If Postal Service management gets its way, computers will list the "work value" of each city block, and managers will combine routes into routes of equal value.

The Postal Service Union opposes the plan because it feels that the capabilities of the individual would be ignored. Presently, each route is matched to an individual, taking into consideration each person's age, sex, and physical condition. "If they implement it beyond the testing cities, I have no alternative but to order a national work stoppage," said James E. Rademacher, president of the National Association of Letter Carriers.

The computer system is advantageous because it will be more economically feasible for management. Some routes could be cut, and letter carriers that would need assistance could be helped by overtime workers.

Rademacher said that his group had no objection to computers as such, and a route evaluation system could be acceptable, he said, "if somehow, you could have consideration for the worker." James R. Braughton, director of delivery services for the U. S. Postal Service, said that fear of robotization was based on misunderstanding, "a lot of it on what role the computer will play."

Office of Science Information Systems Awards $$

Each year, the National Science Foundation's OSIS Awards give financial help to proposed scientific research programs. Last year's awards included some new computer research projects.

One project, proposed by the Interuniversity Communications Council, Inc. in Princeton, will be jointly funded by the Foundation and its Division of Computer Research. It is directed toward gaining a clearer understanding of the implications of network membership upon educational institutions; it seems that economic, political, and organizational problems prevent educational institutions from joining networks linking computer-based activities. The approach is to develop a simulation model for use later in a gaming study involving approximately twenty institutions.

Another award was given to the National Serial Data Program at the Library of Congress for the establishment of an automated national data base on serials in science and technology. When the project is completed, data from any of the world's journals will be available with minimal effort from a possible 50,000 to 60,000 journals. Resources from libraries all over the country are being used.

The University of Pittsburgh's Office of Communications Programs also received an award for the final phase of a 54-month project to develop a Campus-Based Information System (CBIS). CBIS furnishes the faculty and students with information services by providing remote access to computerized files of professional societies, government agencies, commercial organizations, and other universities. The final 18-month phase includes the development of system software to permit university-wide access to data bases through terminals; institutionalization of the administrative relationships of the CBIS with the Library and Computer Center; and the addition of several new data bases.

Dear Mr. Computer:

Forbes' publisher, James Dunn, received this letter from the Plaza Hotel:

"Please pray for us.

"On the first of January of this year we converted our Accounts Receivable system to Electronic Data Processing. In other words, we have surrendered to a computer.

"However, we are determined to have the most personal computer yet employed and have vowed never to hide behind it or use it as an excuse for the human errors to which we are all subject . . . .

"If you have any difficulty at all, please call Ms. Davis. She reports to me, and I whack the computer.

(signed) William E. Pringle
Computer Tamer"

[The salutation on Mr. Dunn's letter was "Dear Mr. Donn:"]

Goodbye to Back-Seat Drivers?

A computerized driving system is being tested in Munich, Germany. A driver identifies his destination upon entering a highway with buried induction loops picking up the information, which is relayed to a central computer which sends back instructions to be followed at the next intersection.

Social Science Literature File

The Institute for Scientific Information in conjunction with Lockheed Information Systems now provides the world's largest social sciences journal literature file. This file gives on-line, interactive, computer searches of the journal literature concerning all social science disciplines published since 1972.

The service, known as SOCIAL SCISEARCH, covers every editorial item ever dealt with in 1200 of the most important social science journals plus 2500 natural and physical science journals. This huge resource center can be searched by using key words, phrases, word-stems, authors, organizations, and citations used in articles as references.

The Lockheed retrieval program, DIALOG, enables subscribers to search the file through two-way communications terminals in their own facilities. This exchange with the computer in Palo Alto, California is conducted in English language statements, and the link can be made with the equivalent of a local phone call.

A Tragedy of Errors

by Susan Hastings

Last year in Florida a man was shot and killed by a state trooper when he parked his car on the shoulder of a highway. In New Hampshire another man was rousted from bed at 4:00 AM and arrested after he had been spotted driving with a defective tail light. Both incidents occurred not because the victims were wanted criminals, but because the people who operated the computerized criminal data bases in their states failed to update information held within those systems.

In December 1975, Frank D. Booth was on the way to his father's funeral when he pulled off the road, apparently to compose himself in the face of his grief. Trooper Robert Rennie, Jr. saw the parked car and radioed an inquiry on Booth's 1974 license plate number to a dispatcher with access to Florida's criminal justice system. When Rennie received a reply which indicated that a car with that number was stolen, he approached it with his .38 caliber pistol loaded and cocked. Seeing Booth reach into his coat and fearing the worst, Rennie fired his gun.

Booth probably would be alive today if officer Rennie had realized that he had received a report about a car stolen in 1971 which just happened to have a license plate with the same number as Booth's 1975 Chrysler. In Florida, vehicle identification is determined not only by tag number, but also by the year the tag was issued. Every year the same numbers are issued at random on plates whose color signifies the year. So-called "hot" numbers are never deleted with the new issuance because a change in color is supposed to negate all old plates.

Yet despite the fact that it takes two criteria—the plate number and the year issued—to correctly identify a vehicle, police can gain access to information in the criminal data base without the second identification criteria. Because the number of the unrecovered 1971 stolen car was still on file, and because the number on Booth's plate coincided with it, Rennie's query produced a reply that convinced him that he was in a dangerous position when he walked up to the man in the car, and he prepared himself for trouble. Officer Rennie was suspended temporarily from the state police pending a coroner's investigation.

When William A. Smith was stopped by Dover, New Hampshire police and issued a defective equipment tag he did not realize that six hours later he would be handcuffed and dragged to police headquarters to be photographed and fingerprinted for an unknown charge. But apparently when the police officer had given Smith his ticket he had wired Smith's name and date of birth to the National Crime Information Center in Washington, D.C. where a "hit" was registered.

According to New Hampshire law, police may hold a suspect for four hours without charging him, and Dover police spent most of that time trying to get Smith to admit being named "Barnes," the surname of a man wanted by New York police who went by the alias "Bill Smith." During the period of the real William Smith's interrogation no attempt was made to verify whether the "Bill Smith" sought by police was still at large, and Smith was refused the opportunity to call upon anyone who could have proven his identity. It was later learned that the man who went by the name "Bill Smith" had been picked up before Smith was even ticketed, but that this information had failed to reach the NCIC wanted persons file.

Although at the time William Smith was so anxious to regain his freedom that he signed a waiver exonerating the police in his false arrest, he is now suing seven policemen for violating his civil rights. If this case goes the way officer Rennie's did however, it is probable that Smith will lose his lawsuit. Six jurors acquitted the trooper on the grounds of justifiable homicide because, according to the Florida Highway Patrol, "all they could go on is what Rennie believed to be the case."

Computerized criminal justice systems have proven themselves to be great aids in law enforcement in this country. But the kind of careless mistakes in updating the systems that led to William Smith's false arrest and Frank Booth's death must be avoided. Police who want to continue to benefit from the data banks will have to learn to use them intelligently and responsibly. The public must also learn to question the misuse of such systems for them to be fully effective and beneficial.

Footnote: Following his acquittal in Florida, Robert Rennie went back on the job as a State Trooper. However, the psychological burden of guilt became too much for him. He subsequently got divorced, quit the State Police and today can only be described as a broken man.

This 'Hit' Not a Miss

ASHKUM, Ill. – A state trooper patrolling the interstate highway near here recently checked the license number of a vehicle apparently abandoned on the shoulder of the road.

A check with the Law Enforcement Agencies Data System (LEADS) and the National Crime Information Center (NCIC) showed the car had been stolen in Cook County and entered into the system earlier that day.

The trooper discovered a young male sleeping in the rear seat and took him into custody.

Further checking revealed the man was wanted for armed robbery and rape of the automobile's owner, and he was turned over to Cook County authorities.

Police in the United Kingdom use a large scale Burroughs B6700 computer to store information for the National Police Computer Network. The central data base has records on vehicle registrations, wanted and missing persons, and criminal records. Hopefully, the system is less prone to human error than the various U.S. systems cited on this page.

Suicide Blamed on Computer

On April 7th of this year, Stephen Karagianis was found hanged in his jail cell. The boy's father blames an apparent computer error for the boy's suicide.

In January, young Karagianis had been arrested for the possession of two marijuana cigarettes, but after Stephen had spent three months in jail, a county judge dismissed the case. Unfortunately, local police never received word of the dismissal over their computer network, and Stephen was again arrested, this time for violation of a parole that did not exist. Despite desperate pleas from the boy and his family, and warnings from Stephen's lawyer that he was seriously depressed and could not cope with confinement, police held the boy overnight. Early the next morning, Stephen was found dead. "You can't trust a boy's life to a computer," John Karagianis said as he tearfully began making funeral arrangements for his son.

put...input/output...in

Corrupting Youth With COBOL

Dear Editor:

I would like to comment on Tom Allen's "Algorithmic Basic" (March-April 77 *Creative Computing).* I agree that BASIC is a very restrictive vehicle for the communication of algorithms, but I believe it would be a disservice to your readers to adopt Allen's COBOL-isms. I myself believe that the structure of a program is made clearest by a pseudo-Algol format. The statement-grouping ability of Algol allows one to discern program structure at a glance without using an excessive number of procedures. A printed page is two-dimensional and Algol can be written so as to take advantage of both dimensions to depict structure. Please do not corrupt the minds of the youth of America (as well as encouraging writer's cramp) by teaching them to think in COBOL.

Professor Alan Filipski
Department of Mathematics
Central Michigan University
Mt. Pleasant, MI 48859

A Fantastic Opportunity for All

Dear Editor:

With regard to your recent letter and shipment of eight (8) "New Creative Computing Catalogues" —

Unfortunately I do not know eight (8) people who are interested in computers (who haven't already subscribed to *Creative*). I don't think I know even one. However I will do my best to carry out the awesome responsibility which was unexpectedly placed on me.

I would like to inquire how it was determined that eight (8) was the correct number of "New Creative Computing Catalogues" to mail each reader. Intensive research indicates that the algorithm: # flyers= (.09059)(log(# readers))(.4(# readers))$^{1/3}$ which yields a result of 7.7926 flyers per reader. This could easily be accomplished by the use of "Fractional Flyers."

I'd also like to take this opportunity to invite you to join Creative Computing Local #109. This small organization has thus far been limited to computer freaks (alias: compulsive programmers, hackers, and computer jocks). However the chapter has been seriously considering admission of people who consider themselves above bombing a large timesharing system on the last day of the semester at 4 PM. Creative Computing Local #109 has no immediate goals but this has not interfered with any of its activities. Hurry, join now, before the entire membership gets bored and he decides to quit! Already, such noted personalities as President Ford, Robert Redford, Isaac Asimov, and Linus Pauling have *not* joined! Noted computer philosopher John Lees has in fact declined nomination! Now is the time!

In closing, I would like to urge you to write again.

Steve North
President
Creative Computing
Local #109
7 Deerhaven Lane
Newfoundland, NJ 07435

Ed. Note: Sounds like an opportunity not to be missed. Dave Ahl and I leaped at the chance to join Local 109! I'd rather not say which way we leaped, however. — BG

SWARMS and EUCHRE

Dear Editor:

In your May-June 1977 issue you published two very imaginative games — SWARMS and EUCHRE. I was able to implement SWARMS on my Compucolor 8001 using color graphics for a continual display of the U.S. map and casualty report in less than 12K bytes of BASIC. The game is well-designed and plays well. The second game, EUCHRE, has an almost unbeatable strategy. I think that a very good player would not be able to beat the computer consistently because the program cheats! First, when the computer decides on trump, it looks at all 12 cards in each hand. Second, when the computer leads a card, it examines all the opponent's playable cards (your board and hand) before making its decision. These two improper strategies could be naively fixed by changing a few FOR loop indices, but this would degrade the computer's playing ability considerably because its decisions would be based on only the exposed cards and not on statistical probabilities or knowledge gained through play. While possibly this particular version of EUCHRE should be abandoned, the game itself is an ideal place for developing probablistic game strategies. The game of EUCHRE is a forerunner of Bridge and is a great deal simpler to implement than Bridge.

Gregory F. Whitten
74 Craigie St., Apt. 10
Somerville, MA 02143

put...input/output...in

A Second Look at Calculators

Dear Editor,

Like E. R. Tufte in *Creative Computing* (May-June 1977, p. 34), I have 5 calculators in the house, 2 of which I consider lemons (though all of them work as advertised), and I have also encountered the feeling of guilt over my cumulative investment (which besides might have bought me a "luxurious" microcomputer system, for over $1270). While his observations on electronic obsolescence are a fair warning to uncreative computing fans or fellow travelers, most of your space could have been used more constructively, to suggest rather the following:

1. Printing is *not* worth it, because one can recall registers or repeat calculations as often as one wants, and one can program "split displays" of several numbers in one display window, or sequential displays of whole arrays, if desired with identifying subscripts flashed briefly before each output. What would you do with all the printout, if it is not labeled, and not on a page you can file or publish, anyway? (Buying a non-programmable with a printer built in is a particularly self-defeating combination, as standards in electronics change fast, and printer mechanics are stagnant and expensive.)

2. A good up-to-date card-programmable calculator is *not* likely to be "purchased with good intentions, and used once or twice," by anyone interested in creative computing. And I have seen players of ready-made games of skill return to Moon Landing or Battle Ships more than twice even on a less convenient key-programmable.

3. Calculators are *not* limited to engineers and business people, as one can see in the HP Users Library, which picked up about 5000 well-documented and distinct programs from owners of just one calculator model, in 3 years. Only about 8% were business, and about 34% engineering oriented. (Incidentally, as a note on software-sharing sociology, the world's most famous minicomputer, the PDP-8, scored less than 1000 contributed programs in the DECUS Library, in 6 years.)

The problem of "compulsive investor's guilt" was solved for me in 2 ways. First, since 1974 my family and I have written up 21 programs we thought noteworthy, and got them all accepted in the internationally accessible HP-65 Users Library, and we have quite a few more in the (now congested) pipeline to the HP-67 Users Library. Secondly my wife, a biologist, did all computing for publication of her master's thesis, and for preparation of her custom-written programs, and enjoying absolute convenience and flexibility of access at her desk, near-perfect efficiency, and not the slightest distraction from hardware reliability problems.

Paradoxically, I don't continue to use any of the "big" programs that were so exciting to write and shake down. Professor Tufte's comparison with Dr. Samuel Johnson's dog walking on its hind legs seems stunningly perceptive in this connection. However, is it really unnatural to use a handy toy to model computational processes on a small scale, for more intimate familiarization? Or to break the tedium of reading a mathematical or technical textbook by breezing through examples which the editor may have found too laborious to check for accuracy? And is it not a stimulating mirror for one's mathematical fitness and quantitative applications ingenuity, independent of academic conventions, if one has occasion to see what turns up, besides games, in a calculator users library? In addition to the more spectacular competitive computing sportsmanship, which is typically elicited by programmable calculators, the keen observer will get a moderate traffic of little cases in daily life, which we still have to learn to recognize as instances of programmable computation; e.g., converting data to percentages which manifestly add up to exactly 100.

Although I got burned twice already, in a way, I frankly look forward to discrete little machines with more capabilities, to get hands-on experience with more challenging mathematical procedures like prediction filtering (applicable in Battle Ships like in Norbert Wiener's anti-aircraft guns), pattern recognition, feature selection, simulation of multivariate random processes, and ultimately, management of my appointments and agenda diary, with the whole show remaining easily portable like a tricorder in STARTREK. Some programs recently completed in shirt-pocket format are Nonlinear Regression, Factor Analysis, Multivariate Analysis of Variance, Exam Grading with Item Analysis, Questionnaire Tabulation, Dynamic Programming, and Master Mind, although not all of these exhibit dazzling performance and range with current equipment.

I will barely take the advice of waiting till the new models appear in the discount stores, and I think the manufacturers are quite justified to plan for "More Power in Your Shirt Pocket or Skirt Pocket."

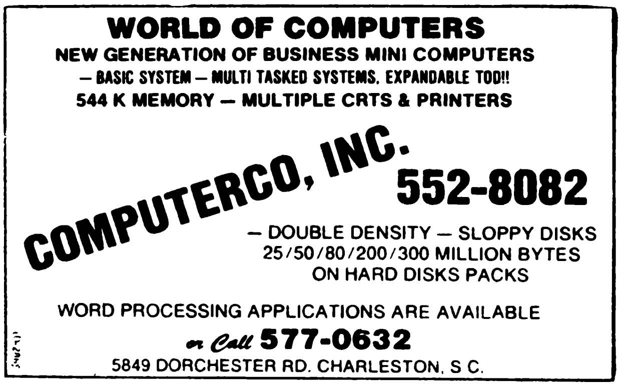
R. Belling
398 Ogden Ave.
Jersey City, NJ 07307

Sloppy Disks

Dear Editor:

Our company is a manufacturing and sales outlet for a new generation of micro and mini computers. We have a franchise aim for a hobby computer store with a strong emphasis on the small business market.

During "Ma Bell's" yellow-page campaign, our company produced two ads showing the internal structure of a business retail and a hobby retail division. As these two ads were given the same address for billing, the printers combined the ads and came up with the enclosed yellow page ad. As you can see, our company is now the proud (?) producers of the only *SLOPPY DISKS* in America for the next year that are double density formated.



WORLD OF COMPUTERS
NEW GENERATION OF BUSINESS MINI COMPUTERS
— BASIC SYSTEM — MULTI TASKED SYSTEMS, EXPANDABLE TOO!!
544 K MEMORY — MULTIPLE CRTS & PRINTERS

COMPUTERCO, INC.
552-8082

— DOUBLE DENSITY — SLOPPY DISKS
25/50/80/200/300 MILLION BYTES
ON HARD DISKS PACKS

WORD PROCESSING APPLICATIONS ARE AVAILABLE
or Call **577-0632**
5849 DORCHESTER RD. CHARLESTON, S C.

We figured that such an engineering feat as *SLOPPY DISKS* would be of interest to your public. Anyone interested in our design structure will be sent an information package.

Douglass C. Boseman
President
ComputerCo, Inc.
5849 Dorchester Rd.
Charleston Hts., SC 29405

■

A for Effort, Zero for Arab

Dear Editor:

On reading the Jan-Feb 1977 issue of *Creative Computing* more carefully, I came upon the item (p 60) titled, "A for Effort, Zero for Arab" which was listed as Anonymous. I believe the author should be given credit for his work.

The piece was written by Mr. William J. Wiswesser and first published in May 1950 under the title, "Prologue: The Empty Column, A Parable about a 'New Notation" of Long Ago."

It was rediscovered and printed again in *Computers and Automation,* January 1970 with a request for the name of the author which was provided in the March 1970 issue.

Mr. Wiswesser's address is 3103 River Road, Reading, Pa. 19605.

I believe his article came about because of his frustration in seeing a delay in acceptance of another "new" notation, which he has developed. It is a compact, linear notation for chemicals which is quite suitable for computer processing, unlike the usual two-dimensional "benzene ring" drawings taught in chemistry courses.

G. Truman Hunter
31 Overlook Drive
Greenwich, CT 06830

Poetry Anyone?

Dear Editor:

I am trying to locate some poetry about computers. If any readers know of any, I would appreciate hearing of it.

D. Van Tassel
Computer Center
University of Calif.
Santa Cruz, CA. 95064

Programming for the Handicapped

Dear Editor:

Enclosed you will find a check and mailing addresses for six subscriptions to *Creative Computing*. These subscriptions are graduation gifts to students of the first graduating class in computer programming for the physically handicapped at Lakeshore.

The course is the third to begin in the nation and the first to be associated with a university (of Alabama in Birmingham). It is a practical, vocationally-oriented course concentrating in COBOL programming with experience in IBM's TSO, JCL, Utilities, and the BASIC programming language.

We thought you might be interested in the existence of the course (and the reason for the subscriptions). We feel that computer programming is an especially appropriate occupation for the physically handicapped as well as an interesting avocation. We believe your magazine will help maintain their interest.

C. Joseph Williams
Lakeshore Rehabilitation Facility
3800 Ridgeway Drive
Birmingham, AL 35209

Endless Repeats

Dear Editor:

Products or quotients which endlessly repeat numbers or patterns of numbers have long intrigued me. Some time ago I gave myself the problem: What are the simplest numerators and denominators whose quotients forever repeat each of the numbers 1 through 9? — and I found, of course, that the desired number, divided by 9, endlessly repeats that number. Thus 7/9 = 7777777... The only difficulty was that 9/9 = 1. Here I had to make an infinitesimal adjustment: 89999999999.../9 = 99999999999....

But just the other week I was reading an essay by Nanekal Senrab, an 11th century Arabic mathematician, wherein he reported that, through sheer serendipity, he discovered that any 2-digit number (other than 99) divided by 99 forever repeats that number. Thus 27/99 = 2727272727...(For 99 divide by three nines instead of two, thus: 99/999 = 099099099099....) For three digit numbers (except 999) divide by 999. Thus 987/999 = 987987987...... The number of digits in both numerator and denominator must match. But this can go on ad infinitum, providing infinite delight.

Why dividing a 2-digit number by 99 repeats that numerator was not clear to me until I realized that dividing by 99 is the same as multiplying by its reciprocal which is 01010101... The reciprocal easily illustrates why the rule is true.

I am happy to share this with your readers.

Lakenan Barnes
115 South Jefferson St.
Mexico, MO 65265

Shorter and Faster ELIZA

Dear Editor:

Steve North's version of ELIZA programmed in MITS BASIC (July/August 1977) is up and running in our ALTAIR. It seems to be interesting to everyone who sees it. One important change though will allow ELIZA to run with nearly 2K bytes less memory: Delete line 90 which allocated array space for strings. These are never used since in MITS BASIC each non-dimensioned string variable and each member of a string array can each contain up to 255 characters.

For use on MITS version 4.0 EXTENDED BASIC, ELIZA will run faster if the following two changes are made:

1. DEFINT A-Z to define all non-string variables as integer and
2. Replacing lines 315-350 with a line incorporating INSTR the string search function
   330 T = INSTR(I$,K$):   IF T THEN S=K: F$=K$: GO TO 365.

Alan R. Miller
Dept. of Metallurgical & Materials Eng.
New Mexico Tech
Socorro, NM 87801

A Plug for PASCAL

Dear Editor:

I am writing in response to the letter by Tom Allen (p 43) in the March-April 1977 issue of your magazine. Mr. Allen's view on the inadequacies of *Basic* and Flowcharting as the standard medium for algorithms in *Creative Computing* is eminently sensible, but his proposal for a *Basic-Cobol* hybrid as such a standard is beyond belief.

There is already a language which expresses algorithms rigorously without the illegibility of *Algol*, and which has the further virtues of begin extremely easy to learn, very powerful, and already implemented. That language is *Pascal*. Consider for example how the main algorithm of Mr. Allen's "Guess a Number" program would look when written in *Pascal:*

```
printinstructions;
repeat getcomputernumber;
  repeat humanguess
  until guesscorrect
until b;
```

Could Mr. Allen really claim that his version expresses the underlying algorithm as clearly as this? Proof to the contrary is to be found in the fact that his version contains no terminating condition; this is not immediately obvious from his program, as it would be if the "b" were omitted from the *Pascal* program.

As *Pascal* interpreters and compilers become available for the 8080, the Z-80 and all the rest later this year, the use of anything but *Pascal* as the standard for personal computing programs will become less and less excusable.

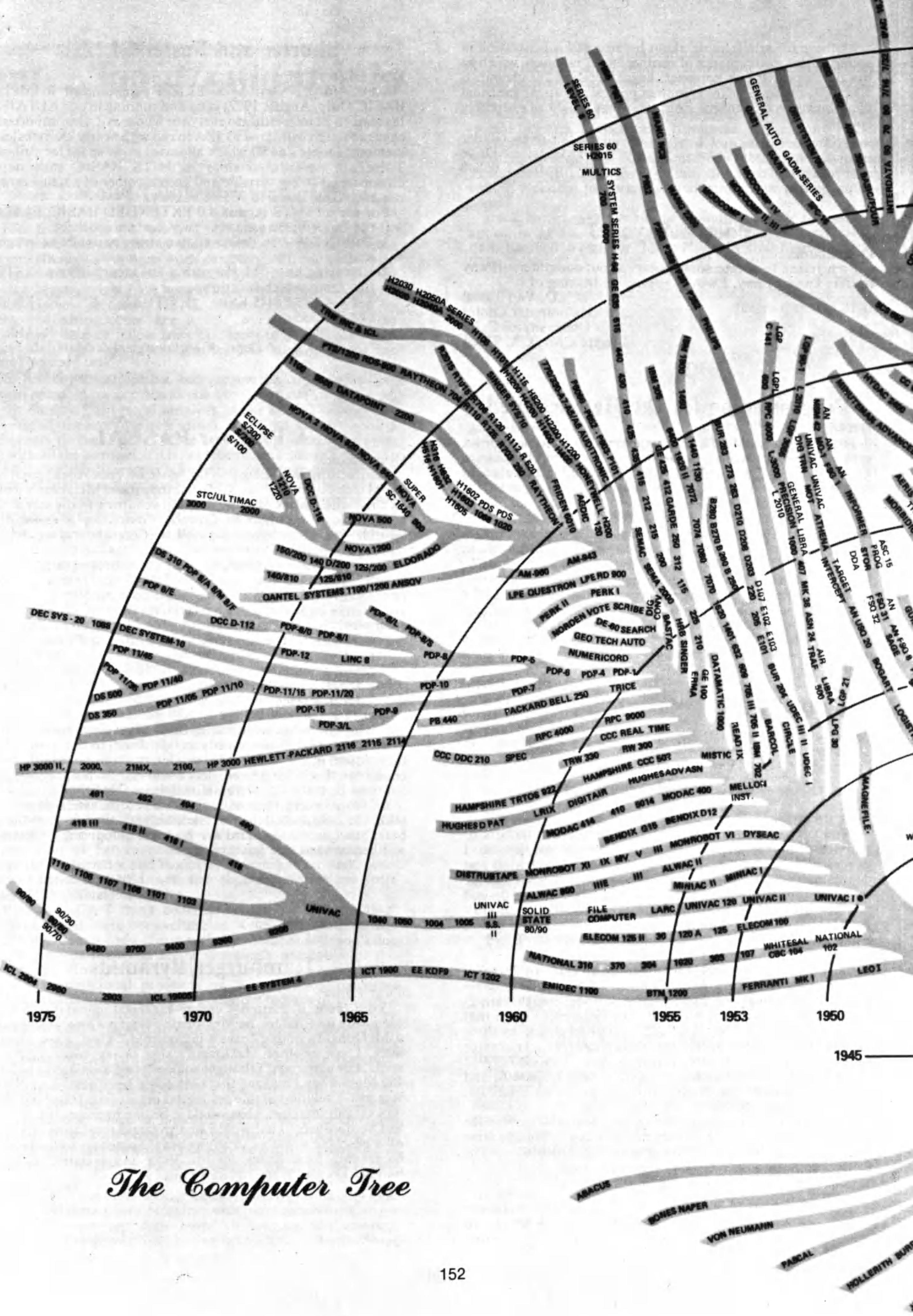
David A. Mundie
104B Oakhurst Circle
Charlottesville, VA 22903

Hamburger Pyramids

Dear Editor:

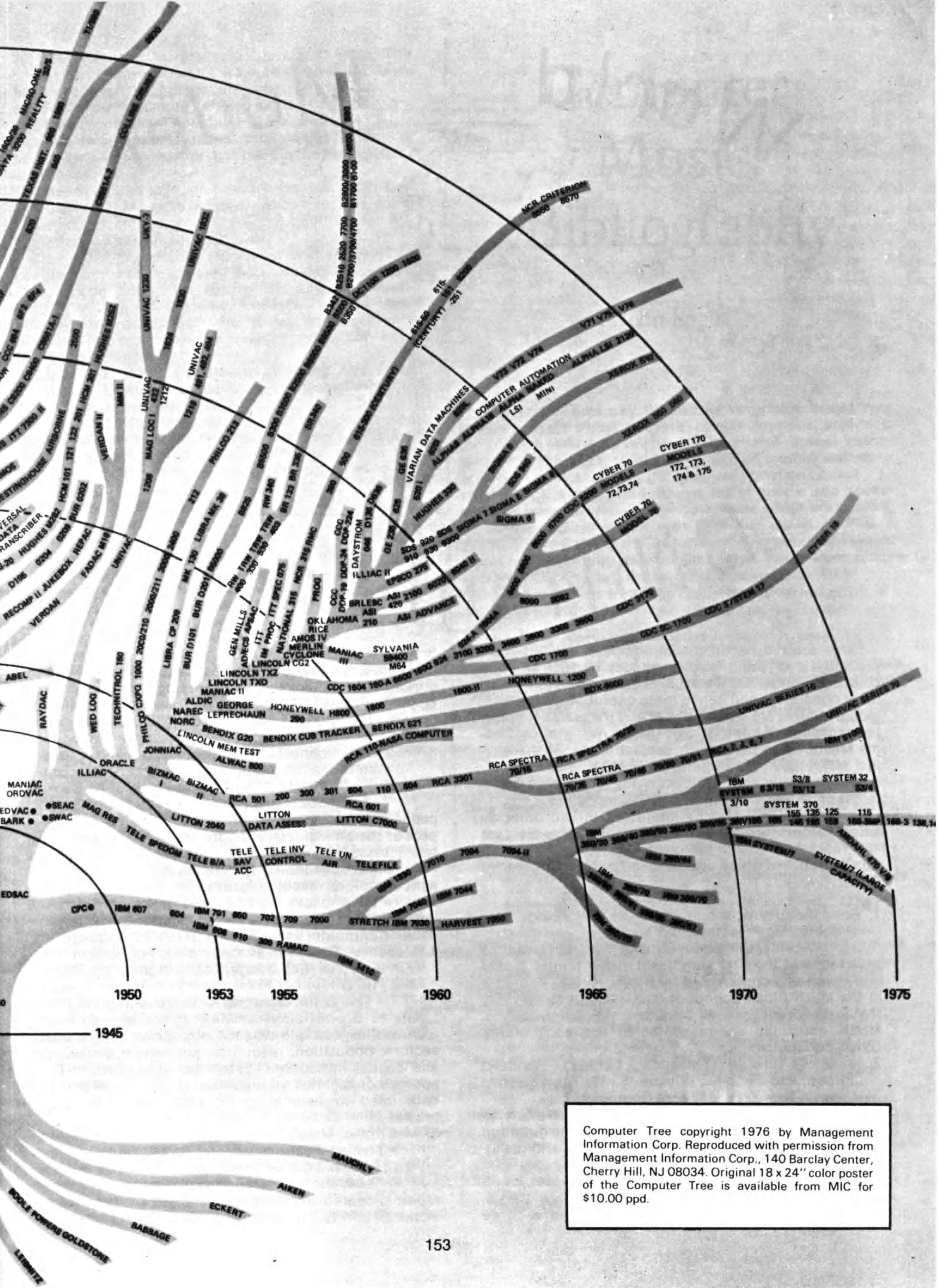
Your cute quotation "under 1,000,000 circulation" was obviously inspired by the McDonald's slogan. Once, I went past a McDonald's billing "over 8 billion sold." Then, a few blocks later, I saw another McDonald's sign saying "over 7 billion sold." For a moment I thought someone had misplaced a billion Big Macs when I realized that both could be correct. And that it was just a billion hamburger lag taking its time travelling two blocks. But imagine, what would it be like to discover 1 billion lost hamburgers? That would be a pyramid measuring 409 feet on a base! Now that's a steaming heap! Someday, someone will be reading back issues, see this letter and exclaim: "Only 8 billion? They're in the trillions now!".

Whimsically,
John Miller
1895 Buck
West Linn, Oregon 97068



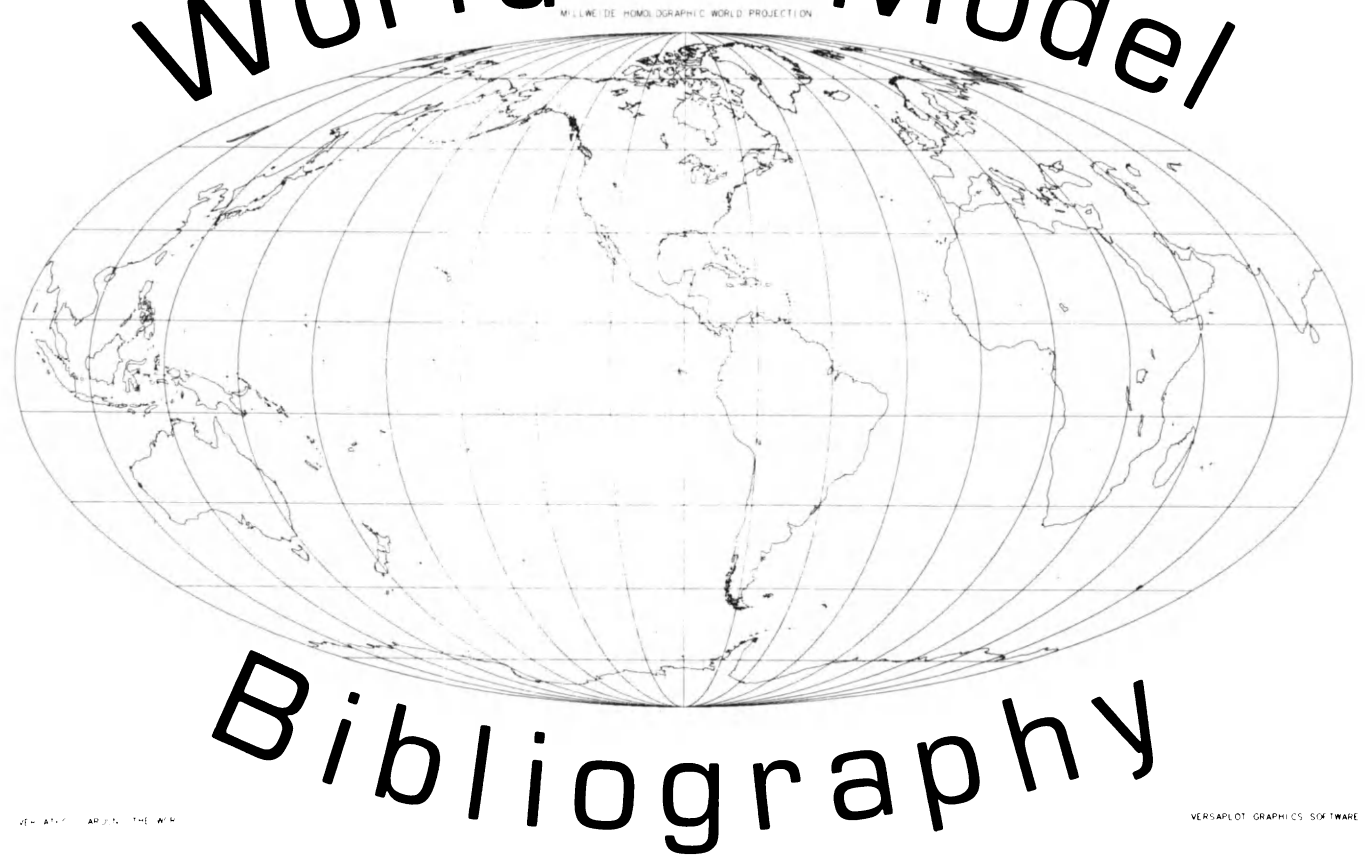
1975
1970
1965
1960
1955
1953
1950
1945
SERIES 60 LEVEL 6
SERIES 60 H2015
MULTICS
SYSTEM SERIES H-64
GENERAL AUTO
GADM-SERIES
INTERDATA
H2030 H2050A SERIES
ECLIPSE S/300 S/200 S/100
NOVA 1210 1220
DCC D-116
SUPER NOVA SC 1640
NOVA 800
NOVA 1200
STC/ULTIMAC 3000 2000
ELDORADO
QANTEL SYSTEMS 1100/1200 ANSOV
PDP 8/E
DCC D-112
DEC SYS - 20
DEC SYSTEM-10
PDP 11/45
PDP 11/40
PDP-12
LINC 8
PDP-8
PDP-5
PDP-6
PDP-4
PDP-1
PDP-10
PDP-7
PDP 11/05
PDP 11/10
PDP-11/15
PDP-11/20
PDP-15
PDP-9
PDP-3/L
DS 500
DS 350
PACKARD BELL 250
PB 440
TRICE
RPC 4000
RPC 9000
CCC REAL TIME
CCC DDC 210
SPEC
TRW 330
RW 300
MISTIC
HP 3000 II, 2000, 21MX, 2100, HP 3000 HEWLETT-PACKARD 2116 2115 2114
HAMPSHIRE TRTOS 922
HAMPSHIRE CCC 507
HUGHES ADV ASN
HUGHES D PAT
LRIX
DIGITAIR
MODAC 414
MODAC 400
MELLON INST.
BENDIX G15
BENDIX D12
DYSEAC
DISTRUSTAPE
MONROBOT XI IX VI V III
MONROBOT VI
ALWAC II
ALWAC 800
ALWAC III E
MINIAC I
MINIAC II
491
492
494
418 III
418 II
418 I
490
418
1110 1108 1107 1106 1101 1102
UNIVAC
1040 1050 1004 1005
UNIVAC III
SOLID STATE 80/90
FILE COMPUTER
LARC
UNIVAC 120
UNIVAC II
UNIVAC I
90/60
90/30
90/70
9480 9700 9400 9300 9200
ELECOM 125 II
ELECOM 120 A
ELECOM 100
WHITESAL CBC 104
NATIONAL 102
NATIONAL 310
ICL 2904
2960
2903
ICL 1900S
EE SYSTEM 4
ICT 1900
EE KDF9
ICT 1202
EMIDEC 1100
BTM 1200
FERRANTI MK I
LEO I
PERK I
PERK II
LPE QUESTRON
LPE RD 900
NORDEN VOTE SCRIBE
DE-60 SEARCH
GEO TECH AUTO
NUMERICORD
AM-960
AM-943
HOLLERITH SINGER
BASTAC
ERMA
GE 100
DATAMATIC 1000
READ IX
BARCOL
CIRCLE
UDEC III
UDEC II
UDEC I
LGP 30
LGP 21
LIBRA 500
AIR TRAF.
ASC 15
PROG
STOR
DDA
TARGET INTERCEPT
FSQ 31
FSQ 32
AN USQ 20
SAGE
BOGART
UNIVAC ATHENA
GENERAL PRECISION L 2010
L-3000
RPC 4000
LGP
C 141
MAGNEFILE
RAYTHEON
FRIDEN 6010
HONEYWELL H200
AUDITRONIC
RAYTHEON 704
PTS/1200 RDS-500 RAYTHEON
DATAPOINT 2200
NOVA 2 NOVA 800 NOVA 840
H1602 PDS PDS
H1603 1068 1020
H1605
ABACUS
BONES NAPER
VON NEUMANN
PASCAL
HOLLERITH

The Computer Tree


1945
1950
1953
1955
1960
1965
1970
1975
LEIBNITZ
BOOLE POWERS GOLDSTONE
BABBAGE
ECKERT
AIKEN
MAUCHLY
EDSAC
EDVAC
BARK
SEAC
SWAC
MANIAC
ORDVAC
ILLIAC
ORACLE
JONNIAC
RAYDAC
WED LOG
TECHNITROL 180
PHILCO CXPQ 1000 2000/210 2000/211
CPC
IBM 607
604
IBM 701
650
702
709
7090
IBM 608
610
305 RAMAC
IBM 1410
IBM 1620
7010
7094
7094-II
IBM 7040
IBM 7044
STRETCH IBM 7030
HARVEST 7950
IBM 360/30
360/40
360/50
360/90
IBM 360/44
IBM 360/70
IBM 360/20
360/195
IBM SYSTEM/7
SYSTEM/7 (LARGE CAPACITY)
AMDAHL 470 V/6
IBM SYSTEM 3/10
S3/8
S3/12
SYSTEM 32
S3/4
SYSTEM 370
155
135
125
115
145
158
168-3
IBM 5100
MAG RES
TELE SPEDOM
TELE B/A
TELE SAV ACC
TELE INV CONTROL
TELE UN AIR
TELEFILE
LITTON 2040
LITTON DATA ASSESS
LITTON C7000
BIZMAC I
BIZMAC II
RCA 501
200
300
301
604
110
RCA 601
RCA 3301
RCA 110-NASA COMPUTER
RCA SPECTRA 70/15
RCA SPECTRA 70/25
RCA SPECTRA 70/35
70/45
70/46
70/55
70/61
RCA 2, 3, 6, 7
UNIVAC SERIES 70
ALWAC 800
LINCOLN MEM TEST
BENDIX G20
BENDIX CUB TRACKER
BENDIX 621
NORC
NAREC
ALDIC
GEORGE
LEPRECHAUN
HONEYWELL 290
H800
1800
1800-II
HONEYWELL 1200
SDX-9000
MANIAC II
LINCOLN TXO
LINCOLN TXZ
LINCOLN CG2
CDC 1604
160-A
924
3100
3200
3400
3600
3800
3300
3500
CDC 1700
CDC 3170
CDC SC-1700
CDC SYSTEM 17
6400
6600
6500
6700
8000
8092
CYBER 70 MODEL 76
CYBER 70 MODELS 72,73,74
CYBER 170 MODELS 172, 173, 174 & 175
CYBER 18
CYCLONE
MERLIN
AMOS IV
RICE
OKLAHOMA
MANIAC III
SYLVANIA S9400
M64
ASI 210
ASI ADVANCE
ASI 420
2100
BRLESC
EPSCO 275
ILLIAC II
DAYSTROM
CCC DDP-19 DDP-24 DDP-224
PROG
NATIONAL 315
NCR 315 RMC
ITT
IM PROC
ITT SPEC 075
GEN MILLS
ADVCS APSAC
RW 400
TRW 330
TRW 530
RW 340
BR 133
BR 335
SR340
SDS 910
920
930
SDS 9300
SIGMA 7
SIGMA 5
SIGMA 8
SIGMA 6
SIGMA 9
SDS 940
XEROX 530
XEROX 550
GE 225
GE 635
625
HUGHES 330
VARIAN DATA MACHINES
520 I
ALPHA 16
ALPHA NAKED MINI
ALPHA LSI 2120
LSI
COMPUTER AUTOMATION
V73
V72
V74
V71
V75
V76
615-50 (CENTURY)
615-200 (CENTURY)
615-151-251
NCR CRITERION
500
B342
B3500
B350
B2510
2520
7700
B2700/3700/4700
B2800/3800
B1700
DC1100
1200
1800
B5500
B6500
UNIVAC 1230
UNIVAC 1206
MAG LOC I
UNIVAC 422
1212
1218
UNIVAC 491, 492, 494
1824
1830
UNIVAC 1832
UKY-7
UNIVAC
PHILCO 212
212
VERDAN II
MM II
VERDAN
FADAC M18
RECOMP II
JUKEBOX
REPAC
D105
0204
0208
BUR 0202
BUR D101
BUR D201
LIBRA CP 209
LIBRA MK 38
MK 130
HUGHES M252
HUGHES M262
HCM 101
121
122
201
HCM 301
2500
AIRBORNE
WESTINGHOUSE
ITT 7300 II
CDC 604
CS200
CS400
CS551A-1
CS551A-2
TEXAS INST.
840
960
1560
320
MICRO-ONE
REALITY
DATA 3200
1600/30
32/5
TI/990
COLLINS C8500
ABEL
DATA TRANSCRIBER

Computer Tree copyright 1976 by Management Information Corp. Reproduced with permission from Management Information Corp., 140 Barclay Center, Cherry Hill, NJ 08034. Original 18 x 24″ color poster of the Computer Tree is available from MIC for $10.00 ppd.

World Model Bibliography


MILLWEIDE HOMOLOGRAPHIC WORLD PROJECTION
VERSAPLOT GRAPHICS SOFTWARE

Large-scale dynamic global computer models have predicted everything from doomsday by 2050 to complete equilibrium. On a smaller scale, models tend to be more consistent and hence more useful. Nevertheless, global models are getting better and there's an increasing exchange of information between rival groups. Dennis Meadows of Dartmouth, project leader of the original MIT study ("The Limits to Growth" was the outcome), has prepared a bibliography of sources of interest to people involved with global modeling.

The layman will also be interested in two other articles, "Doomsday, Says MIT Computer, May Be Just 100 Years Away," and "Dynamic Modeling Using Fortran IV," both of which appeared in *The Best of Creative Computing, Vol. 1.*

1. Forrester, Jay W. *World Dynamics.* Cambridge, MA: Wright-Allen Press, Inc., 1971.

World2, the model on which *World Dynamics* is based, was the first significant global model. It was conceived and constructed by Professor Jay Forrester at the M.I.T. Sloan School of Management to serve as the basis for a two-week seminar on global problems. Model equations are presented in the book in the DYNAMO language.

2. Meadows, Donella H.; Meadows, Dennis L.; Randers, Jorgen; and Behrens, William W., III. *The Limits to Growth.* New York: Universe Books, 1972.

Limits was the result of a two-year effort by ten scientists and students at M.I.T. under the direction of Dennis and Donella Meadows to extend and disaggregate the model originally presented by Jay Forrester. While the resulting publication provides much more empirical data in support of the general thesis, the conclusions remain essentially the same as those first sketched out in *World Dynamics.* This text presents a comprehensive verbal description of the important causal mechanisms governing growth and collapse, but it does not present the complete equations required to reproduce the detailed results. Those are included in reference 4 below.

3. Meadows, Dennis L., and Meadows, Donella H., eds. *Toward Global Equilibrium: Collected Papers.* Cambridge, MA: Wright-Allen Press, Inc., 1973.

This book is an unedited collection of thirteen papers, each commissioned to explore a specific aspect of the Limits to Growth project at M.I.T. There are two introductory papers by Forrester and Meadows, seven papers describing specific models of population, agricultural, or resource issues, and four reports that explore the implications of growth limits for different institutions. The seven modeling papers each contain detailed computer listings in the DYNAMO language.

4. Meadows, Dennis L. et al. *Dynamics of Growth in a Finite World.* Cambridge, MA: Wright-Allen Press, Inc., 1974.

This is the technical report resulting from the Limits to Growth effort at M.I.T. It contains detailed technical expositions on each of the five world model sectors: population, resources, agriculture, pollution, and capital investment. Extensive references to the sources of theories incorporated in the model and to data used in estimating its coefficients are also provided.

5. Mesarovic, Mihajlo, and Pestel, Eduard. *Mankind at the Turning Point.* New York: E.P. Dutton and Co., Inc., Reader's Digest Press, 1974.

Mankind at the Turning Point is the general report prepared from results obtained by two teams working jointly at Hannover, Germany, and Case

Western Reserve University in Cleveland, Ohio. The basic conclusions of *Limits* are reaffirmed in this text, but the regional disaggregation employed by Mesarovic and Pestel permits more detailed exploration of regional interactions than could be done with the M.I.T. models. This report does not include the equations necessary to reproduce its runs nor are they available in any other form.

5a. Mesarovic, Mihajlo, and Pestel, Eduard. *Multilevel Computer Model of World Development System.* 6 vols. Luxembourg: International Institute for Applied Systems Analysis, 1974.

This six-volume report set was released in conjunction with a presentation of the Mesarovic-Pestel model before the staff of the International Institute for Applied Systems Analysis, but the complicated input/output software of the model prevents even the information in the detailed exposition from being implemented.

6. Linneman, Hans. *Population Doubling and Food Supply.* Amsterdam: Economic and Social Institute of the Free University, 1974.

The effort of this Dutch team was undertaken in order to determine the investment and social implications of a doubling world population. To narrow the focus of the study, exclusive concern was given to the question of nutrition. The analysis was based in part on simulation models, the equations for which are not given in this document.

7. Herrera, Amilcar O. et al. *Catastrophe or New Society? A Latin American World Model.* Ottawa: International Development Research Centre, 1976.

The Bariloche model was undertaken by a team of social scientists at the Bariloche Foundation in South America with support from the International Development Research Council in Canada. Initially conceied as a criticism and rebuttal of the Limits to Growth thesis, this model is essentially normative in concept and provides some description of those policies which would have to be employed to meet the basic needs of the world's population by the year 2000.

8. Leontieff, Wassily; Carter, Anne; and Petri, Peter. *The Future of the World Economy: A Study on the Impact of Prospective Economic Issues and Policies on the International Development Strategy.* New York: United Nations, 1976.

The Leontieff model was commissioned by the United Nations. It employs an input/output matrix with exogenous population variables and a rather scanty resource and pollution sector to examine interregional trade flows over the next several decades.

9. Kahn, Herman; Brown, William; and Martel, Leon. *The Next 200 Years.* New York: William Morrow and Company, Inc., 1976.

This volume is part of an ongoing study by Herman Kahn and his associates within the Hudson Institute to chart out some areas of the future. The work is not based on any formal computer model, nor are the detailed underlying assumptions leading to the authors' conclusions actually specified or incorporated into the text.

10. Tinbergen, Jan. *Reshaping the International Order.* Edited by Antony J. Dolman. New York: E.P. Dutton and Co., Inc., 1976.

RIO is the description of a study undertaken to investigate narrowing the gap between the rich and the poor countries of the world. Although the organizers did not use a formal computer model, they did develop, through meetings of international committees, several schemes for economic equity. The thrust of the study is distribution and self-sufficiency. ■

Computer Music Bibliography

John Snell

Digital circuits may be used for controlling analog synthesizers, direct digital synthesis, composing music, analyzing (or tracking several parameters of) traditional musical instruments and the voice, spacial movement of sounds, and processing of musical sounds (filtering, reverberation, choral effects, etc.). I hope the following list of articles and books will help some of you to develop systems which are capable of making music enjoyable even by master musicians. This list is relatively short, and includes only a "taste" of relevant topics not specifically about digital music. For a more comprehensive, well-organized listing see the bibliography from *Electronotes* (a fine electronic music periodical edited by Bernie Hutchins).

J. Allen, "Computer Architecture for Signal Processing," *IEEE*, Vol. 63, No. 4; April, 1975.

W. Apel, *Havard Dictionary of Music*, Halliday Lithograph, 1969.

J. Backus, *The Acoustical Foundations of Music*, W.W. Norton, 1969.

J.K. Baker, "The DRAGON System—An Overview," *IEEE*, Vol. ASSP-23, No. 1; Feb., 1975.

S.C. Bass, and B.J. Leon, "Designers' Guide to Digital Filters," *EDN*, Jan. - June, 1974 (6 issues).

J.W. Beauchamp & H. Von Foerster, editors, *Music by Computer*, Wiley, 1969.

J.W. Beauchamp, "Analysis and Synthesis of Cornet Tones Using Nonlinear Interharmonic Relationships," *JAES*, Vol. 23, No. 10; 1975.

J.W. Beauchamp, "A Computer System for Time-Vatiant Harmonic Analysis and Synthesis of Musical Tones," *Music by Computers*, 1969.

J.W. Beauchamp, "Additive Synthesis of Harmonic Musical Tones," *JAES*, Vol. 14, No. 4, p. 332; 1966.

L. Beranek, "Digital Synthesis of Speech and Music," *IEEE Transactions on Audio*, Vol. AU-18, No. 4; Dec., 1970.

S. Bertram, "Frequency Analysis Using the Discrete Fourier Transform," *IEEE Transactions on Audio and Electroacoustics*, Vol. AU-18, No. 4; Dec., 1970.

B.A. Blesser, K. Baeder, & R. Zaorski, "A Real-Time Digital Computer for Simulating Audio Systems," *JAES*, Vol. 23, No. 9, p. 698; 1975.

B.A. Blesser, "An Investigation of Quantization Noise," *JAES*, Vol. 22, No. 1, p. 20; 1974.

B.A. Blesser, & F. Lee, "An Audio Delay System Using Digital Technology," *JAES*, Vol. 19, No. 5, p. 393; May, 1971.

E. Blom, editor, *Grove's Dictionary of Music and Musicians*, St. Martin's Press.

D. Bohn, editor, *Audio Handbook*, National Semiconductor Corp., Santa Clara, Cal.; 1976.

Research Engineer, People's Computer Co. Box 310, Menlo Park CA 94025; 415 323-3111

Copyright © 1977 by *Dr. Dobbs Journal* P.O. Box 310, Menlo Park, CA 94025.
Reprinted by permission.

R.W. Burhams, "Digital Tone Synthesis," *JAES*, Vol. 19, No. 8; Sept., 1971.

R.W. Burhams, "Single-Bus Keyboard Control for Digital Musical Instruments," *JAES*, Vol. 19, No. 10, p. 865; Nov., 1971.

R.W. Burhams, "Simplified Educational Music Synthesizer," *JAES*, Vol. 19, No. 2, p. 127; Feb., 1971.

R.P. Ceely, "A Composer's View of MITSYN," *AES Preprint No. 811 (M-1)*, Oct., 1971.

H. Chamberlin, "Fourier Series Waveform Generator, Part 1," *EN*, Vol. 5, No. 39, p. 2; May, 1974.

J.M. Chowning, J.M. Grey, L. Rush, & J.A. Moorer, *Computer Simulation of Music Instrument Tones in Reverberant Environments*, Stanford University Dept. of Music Report No. STAN-M-1; June, 1974.

J.M. Chowning, "The Simulation of Moving Sound Sources," *JAES*, Vol. 19, No. 1, p. 2; Jan., 1971.

J.M. Chowning, "The Synthesis of Complex Audio Spectra by Means of Frequency Modulation," *JAES*, Vol. 21, No. 7, p. 526; Sept., 1973.

J.M. Chowning, "Stanford Computer Music Project," *NW*, No. 1; 1972.

W. Cochran, J. Cooley, D. Favin, H. Helms, R. Kaenel, W. Lang, G. Maling Jr., D. Nelson, C. Rader, P. Welch, "What is the Fast Fourier Transform," *IEEE*, Vol. 55, No. 10, p. 1664; Oct., 1967

R. Crochiere, & P. Penfield, Jr., "On the Efficient Design of Bandpass Digital Filter Structures," *IEEE*, Vol. ASSP-23; 1975.

M.G. Crosby, "Carrier and Side-Frequency Relations with Multi-Tone Frequency or Phase Modulation," *RCA Review*, Vol. 3; 1938.

D.R. Curtis, "A Monolithic Voltage-Controlled Amplifier Employing Log-Antilog Techniques," *JAES*, Vol. 24, No. 2; March, 1976.

D. De Kold, "Binary Division Produces Harmonic Frequencies," *Electronics*, Dec., 1972.

J.L. Divilbliss, "The Real-Time Generation of Music with a Digital Computer," *JMT*, No. 8, p. 99; 1964.

J. Dubnowski, R. Schafer, & L.R. Rabiner, "Real-Time Digital Hardware Pitch Detector, *IEEE*, Vol. ASSP-24, No. 1; Feb., 1976.

J.L. Flanagan & L.R. Rabiner, editors, *Speech Synthesis*, Dowden, Hutchinson, & Ross; 1973.

A. Forte, *Tonal Harmony in Concept and Practice*, Holt, Rinehart, & Winston; 1962.

N.V. Franssen & L. Van der Peet, "Digital Tone Generation for a Transposing Keyboard Instrument," *Philips Tech. Rev.*, No. 31, p. 354; 1970.

M.D. Freedman, "Technique for Analysis of Musical Instrument Tones," Ph.D. dissertation, University of Illinois, Urbana; 1965.

M.D. Freedman, "Analysis of Musical Instrument Tones," *JASA*, No. 41, p. 793; 1967.

M.D. Freedman, "A Method for Analyzing Musical Tones," *JAES*, Vol. 16, No. 4, p. 419; Oct., 1968.

S. Freeny, "Special-Purpose Hardware for Digital filtering," *IEEE*, Vol. 63, No. 4; April, 1975.

R. Gabel, "A Parallel Arithmetic Hardware Structure for Recursive Digital Filtering," *IEEE*, Vol. ASSP-22; Aug., 1974.

J. Gabura & G. Ciamaga, "Computer Control of Sound Apparatus for Electronic Music," *AES preprint No. 520*; 1967.

S. Gill, "A Technique for the Composition of Music in a Computer," *Computer Journal*, Vol. 6, No. 2, p. 129; July, 1963.

J.M. Grey, *An Exploration of Musical Timbre*, Stanford University Dept of Music Report No. STAN-M-2; Feb., 1975.

P. Grogono, "MUSYS: Software for an Electronic Music Studio," *Software—Practice and Experience*, No. 3, p. 369; 1973.

H.L.F. Helmholtz, *On the Sensations of Tone as a Physiological Basis for the Theory of Music*, translation—originally written in German in 1863—and additions by A.J. Ellis, Dover, N.Y.; 1954.

W. Henke, *Multiple Interactive Tone Synthesis System*, Research Laboratory of Electronics, M.I.T.; Oct., 1971.

L. Hiller & L. Isaacson, "Musical Composition with a High-Speed Digital Computer," *JAES*, Vol. 6, p. 154; July, 1958.

L. Hiller, "Computer Music," *Scientific American*, p. 109; Dec., 1959.

L. Hiller, "A Review of Decca Recording DL-9103, 'Music from Mathematics'," *IEEE*, Vol. 51, No. 3, p. 538; March, 1963.

L. Hiller, "Musical Applications of Electronic Digital Computers," *Gravesano Review*, pp. 26-28; Nov., 1965.

C. Hovey & D. Seamans, "A Polyphonic Keyboard for a Voltage-Controlled Music Synthesizer," *JAES*, Vol. 23, No. 6, p. 459; 1975.

K. Huehne, "Programmable ROM's Offer a Digital Approach to Waveform Synthesis," *EDN*, Aug., 1972.

B. Hutchins, "B-004 Sourcelist of Electronic Music and Musical Engineering," *EN*, 1975.

B. Hutchins, *Musical Engineer's Handbook*. (*EN* address) 1975.

B. Hutchins, "The Frequency Modulation Spectrum of an Exponential Voltage-Controlled Oscillator," *JAES*, Vol. 23, No. 3, p. 200; 1975.

B. Hutchins, "Application of a Real-Time Hadamard Transform Network to Sound Synthesis," *JAES*, Vol. 23, No. 7, p. 558; 1975.

B. Hutchins, "Digital Harmonics 1, Waveshaping 2," *EN*, Vol. 2, No. 12, p. 5; Sept., 1972.

B. Hutchins, T. Mintner, & C. Anderton, "Digital Harmonics 2," *EN*, Vol. 2, No. 13, p. 12; Oct. 1972.

B. Hutchins, "Analog Delay for Musical Engineering," *EN*, Vol. 7, No. 56; Aug., 1975.

IEEE, Vol. 63, No. 4, special issue on digital signal processing; April, 1975.

F. Itakura, & S. Saito, "Digital Filtering Techniques for Speech Analysis and Synthesis," in Conf. Rec., 7th Int. Congr. Acoustics, Paper 25C No. 1; 1971.

E.V. Jansson, & J. Sundberg, "Long-Time-Average-Spectra Applied to Analysis of Music," *Acustica*, Vol. 34, No. 1; Nov., 1975.

L.B.W. Jolley, *Summation of Series*, Dover, N.Y.; 1961.

P. Knowlton, "Capture and Display of Keyboard Music," *Datamation*, P. 56; May, 1972.

S. Kriz, "A 16-Bit A-D-A Conversion System for High Fidelity Audio Research," IEEE Symposium on Speech Recognition, p. 278; April, 1974.

P. Lehman, "Harmonic Structure of the Tone of the Bassoon," *JASA*, Vol. 36, No. 9, p. 1649; Sept., 1964.

V. Lesser, D. Fennell, L. Erman, & R. Reddy, "Organization of Hearsay II Speech Understanding System," *IEEE*, Vol. ASSP-23, No. 1; Feb., 1975.

T. Letowski, "Difference Limen for Nonlinear Distortion in Sine Signals and Musical Sounds," *Acustica*, Vol. 34, No. 2; Dec., 1975.

H. Lincoln, editor, *The Computer and Music*, Cornell University Press; 1970.

J. Link, Theory and Tuning: *Aron's Meantone Temperament and Marpsurg's Temperatment "I"*, Tuners Supply Co., Boston; 1969.

B. Lin, & A. Peled, "A New Hardware Realization of High Speed Fast Fourier Transformers," *IEEE*, Vol. ASSP-23, No. 6; Dec., 1975.

H.C. Longuet-Higgins, & M.J. Steedman, *On Interpreting Bach in Machine Intelligence VI;* B. Meltzer, & D. Michie, editors, Edinburgh University Press, Edinburgh; p. 221; 1971.

D. Luce, "Description of a Real-Time Multipartial Waveform Analyzer-Synthesizer," AES preprint No. 611; 1968.

D. Luce, "Dynamic Spectrum Changes of Orchestral Instruments," *JAES*, Vol. 23, No. 7; 1975.

D. Luce, & M. Clark, "Physical Correlates of Brass Instrument Tones," *JASA*, Vol. 42, p. 1232; 1967.

M. Mathews, *The Technology of Computer Music*, M.I.T. Press, Boston; 1969.

M. Mathews, et. al., "Computers and Future Music," *Science*, Vol. 183, No. 4122; Jan. 25, 1974.

N. Mathews, "The Digital Computer as a Musical Instrument," *Science*, Vol. 142, p. 553; 1963.

M. Mathews, & J. Kohut, "Electronic Simulation of Violin Resonances," *JASA*, No. 53, p. 1620; 1973.

C.A. McGonegal, L.R. Rabiner, & A.E. Rosenberg, "A Semi-automatic Pitch Detector," *IEEE*, Vol. ASSP-23, No. 6; Dec., 1975.

N.J. Miller, "Filtering of Singing Voice Signal from Noise by Synthesis," Ph.D. thesis; University of Utah, Dept. of Computer Science; May, 1973.

N.J. Miller, "Pitch Detection by Data Reduction," *IEEE*, Vol. ASSP-23, No. 1, pp. 72-78; Feb., 1975.

F.R. Moore, & M. Mathews, "Grove—A Computer Program for Real-Time Music and Sound Synthesis," *Communications of the ACM*, Vol. 13, No. 12; Dec., 1970.

J.A. Moorer, "The Optimum Comb Method of Pitch Period Analysis of Continuous Digitized Speech," *IEEE*, Vol. ASSP-22, No. 5, p. 330; Oct., 1974.

J.A. Moorer, "Music and Computer Composition," *Communications of the ACM*, Vol. 15, No. 2; Feb., 1972.

J.A. Moorer, *On the Segmentation and Analysis of Continuous Musical Sound by Digital Computer*, Stanford University Dept. of Music Report No. STAN-M-3; May, 1975.

J.A. Moorer, *The Synthesis of Complex Audio Spectra by Means of Discrete Summation Formulae*, Stanford University Dept. of Music Report No. STAN-M-5; Dec., 1975.

J.A. Moorer, *On the Loudness of Complex Time-Variant Tones*, Stanford University Dept. of Music Report No. STAN-M-4; Feb., 1975.

NUMUS West, No. 4; 1973. (computer music issue)

T.H. O'Beirne, "Computer Program Which Plays Music by Microtones," *Computer Journal*, Vol. 13, No. 4, p. 350; 1970.

A.V. Oppenheim, & R.W. Schafer, *Digital Signal Processing*, Prentice Hall; 1975.

W. Piston, *Counterpoint*, W.W. Norton Publishers.

W. Piston, *Orchestration*, W.W. Norton Publishers.

G. Plitnik, & W. Strong, "Digital Filter Technique for Synthesis of Bassoon Tones," *JASA*, No. 47, p. 131; 1970.

R. Plomp, "Timbre as a Multidimensional Attribute of Complex Tones," *Frequency Analysis and Periodicity Detection in Hearing*, edited by R. Plomp & G.F. Smoorenburg; A.W. Sijthoff, Leiden; 1970.

L. Rabiner & B. Gold, *Theory and Applications of Digital Signal Processing*, Prentice-Hall, N.J.; 1975.

L. Rabiner & C. Rader, editors, *Digital Signal Processing*, IEEE Press; 1972.

J.C. Risset & M. Mathews, "Analysis of Musical Instrument Tones," *Physics Today*, Vol. 22, No. 2, p. 23; 1969.

J.C. Risset, "Computer Study of Trumpet Tones," Bell Telephone Lab, Murray Hill, N.J.; 1966.

P. Samson, "Digital Signal Synthesizer," Systems Concepts, San Francisco, Cal.

P. Samson, "Digital Synthesizer—Brief Functional Description," Systems Concepts, San Francisco, Cal.

P. Samson, "Digital Synthesizer—Programming Specifications," Systems Concepts, San Francisco, Cal.

S. Saunders, "FM The Easy Way," Carnegie Mellon University Computer Science Dept & Xerox PARC; 1974.

R.A. Schaefer, "Digital Generation of Equal Temperament," *IEEE*, Vol. AASP-23, No. 4; Aug., 1975.

M. Schroeder, "Natural Sounding Artificial Reverberation, *JAES*, Vol. 10, p. 219; 1962.

M. Schroeder, "Models of Hearing," *IEEE*, Vol. 63, No. 9; Sept., 1975.

R. Schumacher, "Self-Sustained Musical Oscillators," paper delivered at the 91st meeting of the Acoustical Society of America; April, 1976.

J. Sennott, "Efficient Digital Conversion of Audio Signals," AES preprint No. 970 (M-4); Sept., 1974.

W. Slawson, Syntal II; *A Speech Oriented Computer Synthesizer Revisited*, Dept. of Music, University of Pittsburgh, Pa.

W. Slawson, "Vowel Quality and Musical Timbre as Functions of Spectrum Envelope and Fundamental Frequency," *JASA*, No. 43, p. 87; 1968.

G.W. Small, "Synthesis of the Musical Scale Using Non-integer Frequency Division," *JAES*, Vol. 21, No. 4, p. 261; May, 1973.

G.W. Small, "Rate-Feedback Binary Counters in Musical Scale Generation," *JAES*, Vol. 21, No. 9, p. 702; Nov., 1973.

L. Smith, "Score—A Musician's Approach to Computer Music," *JAES*, Vol. 20, No. 1; Jan./Feb., 1972.

L. Smith, "Editing and Printing Music by Computer," *JMT*, p. 292; Fall, 1973.

S. Smith, "Computer Music in 1972," *Computers and Automation*, Oct., 1972.

G. Steinke, "Experimental Music with the Subharchord Subharmonic Sound Generator," *JAES*, Vol. 14, No. 2, p. 140; April, 1966.

S.S. Stevens & H. Davis, *Hearing—Its Psychology and Physiology*, Wiley, N.Y.; 1938.

T.G. Stockham, Jr., "A-D and D-A Converters: Their Effect on Digital Audio Fidelity," AES Preprint No. 834 (D-1), Oct., 1971.

W. Strong & M. Clark, "Synthesis of Wind-Instrument Tones," *JASA*, Vol. 41, p. 39; 1967.

W. Strong, & M. Clark, "Perturbations of Synthetic Orchestral Wind-Instrument Tones," *JASA*, No. 41, p. 277; 1967.

R. Talambiras, "Digital-to-Analog Converters: Some Problems in Producing High Fidelity Systems," *Computer Design*, Vol. 15, No. 1, p. 63; Jan., 1976.

J.C. Tenny, "Sound Generation by Means of a Digital Computer," *JMT*, No. 7, p. 24; 1963.

H.M. Tremaine, *Audio Cyclopedia*, Sams, N.Y.; 1969.

D.E. Walker, "The SRI Speech Understanding System," *IEEE*, Vol. ASSP-23, No. 5, Oct., 1975.

L. Wedin & G. Goude, "Dimensional Analysis of the Perception of Instrumental Timbre," *Scand. Jour. Psych.*, No. 13, P. 228; 1972.

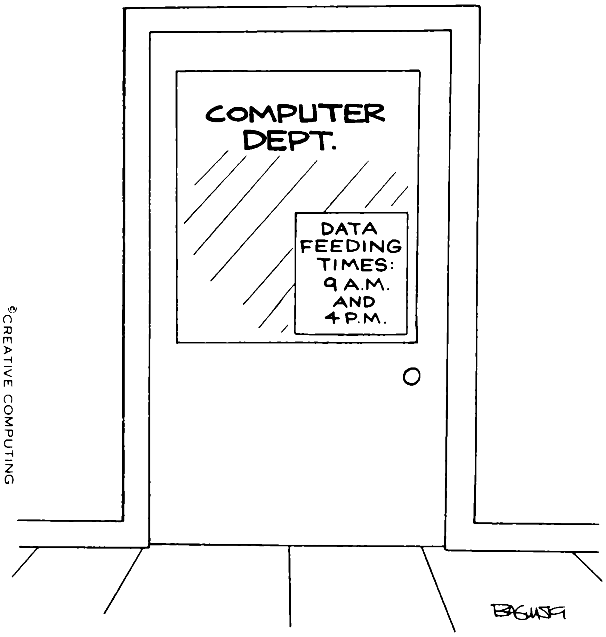
S. White, "On Mechanization of Vector Multiplication," *IEEE*, Vol. 63, No. 4; April, 1975.

T. Winograd, "Linguistics and the Computer Analysis of Tonal Harmony," *JMT*, Vol. 12, No. 1, p. 2; 1968.

J. Zingheim, "Introduction to Computer Music Techniques," *EN*, Vol. 6, No. 42, p. 1; Aug., 1974.

ABBREVIATIONS USED, AND ADDRESSES

ACM	Association for Computing Machinery
	Acustica, S. Hirzel, Stuttgart 1, Birkenwaldstr, 44, Postfach 347, Germany.
AES	Audio Engineering Society
ASSP	*IEEE* Transactions on Acoustics Speech and Signal Processing
	Communications of the ACM 1133 Avenue of the Americas, NYC 10036
	Computer Design Box A, Winchester MA 01890
	Computer Journal British Computer Society 29 Portland Place London, England W1N 4AP
	Computers and Automation [now called *Computers and People*] Berkeley Enterprises, Inc. 815 Washington St, Newtonville MA 02160
	Datamation Technical Publishing Co. 1301 South Grove Ave., Barrington IL 60010
ED	*Electronic Design* 50 Essex St., Rochell Park NJ 07662
EDN	*EDN Magazine* 270 St. Paul St., Denver CO 80206
EN	*Electronotes* 203 Snyder Hill Rd., Ithaca NY 14850
IEEE	*Institute of Electrical and Electronics Engineers* 345 E 47 St., NYC 10017
JAES	*Journal of the Audio Engineering Society* 60 E 42 St Lincoln Blg, Room 929 NYC 10017
JASA	*Journal of the Acoustical Society of America* American Institute of Physics 335 E 45 St., NYC 10017
JMT	*Journal of Music Theory* Yale School of Music Yale University New Haven CT 06520
NW	*NUMUS West* Box 135, Mercer Island WA 98040
	Perspectives of New Music Box 231, Princeton NJ 08540
	Science American Association for the Advancement of Science 1515 Massachusetts Ave. N.W. Washington DC 20005
	Scientific American 415 Madison Ave., NYC 10017


COMPUTER DEPT.
DATA FEEDING TIMES: 9 A.M. AND 4 P.M.

©CREATIVE COMPUTING

Electronic and Computer Music

Reprinted below is the list of currently-available electronic and computer music tapes and records as listed in the January 1977 *Schwann-1 Catalog.*

Key:	
Q	Quad record
▲	8-track tape
Q▲	Quad-8 tape
•	Cassette Tape

Label abbreviations.

Adv.	Advance
Ang.	Angel
At.	Atlantic
Can.	Candide
Col.	Columbia
DG	Deutsche Grammophon
Ev.	Everest
Fin.	Finnadar
Folk.	Folkways
GC	Golden Crest
Lyr.	Lyrichord
Main.	Mainstream
Mer.	Mercury
None	Nonesuch
Odys	Odyseey
Ser.	Serenus
Van.	Vanguard
West.	Westminster

Listings below give complete contents when the entire record contains electronic music. A reference to the listing under the composer's name in the Composer Section is made when only a part of the contents of the record falls into this category. Performers' names are in italics.

Adv. **S-8—MAXFIELD:** Pastoral Symphony (1960); Bacchanale (1963); Piano Concert For David Tudor (1961); Amazing Grace (1960)
Adv. **16—BUDD:** Oak of Golden Dreams (1970); Coeur D'Orr (1969)
Ang. **S-36042;** ●4XS-36042—**WHITE:** Short Circuits Moog synthesizer & other electronics
At. **S-1576—MIMAROGLU:** Sing Me a Song of Songmy—Hubbard, Reciters, Chorus, String Orch., Organ, Mimaroglu, Synthesized & Processed Sounds
Can. **31001—STOCKHAUSEN:** Prozession for Tamtam, Viola Elektronium, Piano, Filters & Potentiometers (1967)
Can. **31025—SCHAEFFER:** Objets liés; **MACHE:** Terre de feu, 2nd version; **PHILIPPOT:** Etude III; **BAYLE:** Oiseau Chanteur; **FERRARI:** Tête et queue du dragon; **MALEC:** Dahovi; **PARMEGLIANI:** Danse
Capra **1201**—*See Duckworth, Strange, Wilson*
Col. **MS-6146**—*See Varèse*
Col. **MS-6362**—*See Varèse*
Col. **MS-6566—AREL:** Stereo Electronic Music No. 1 (1964); **BABBITT:** Composition for Synthesizer (1964); **DAVIDOVSKY:** Electronic Study No. 1 (1964); **EL-DABH:** Leiyla and the Poet (1962); **LUENING:** Gargoyles (1962); **USSACHEVSKY:** Creation—Prologue (1962)—*Col.-Princeton Elec.*
Col. **MS-7051—BABBITT:** Ensembles for Synthesizer; **CAGE:** Variations II (1961)—*Tudor;* **POUSSEUR:** Trois visages de Liège (1961)
Col. **MS-7139**—*See Brown, Bussotti, Cage*
Col. **MS-7194; MQ-31018(Q);** ▲18-11-0092; Q▲MAQ-31018; ●18-11-0092—**SWITCHED-ON BACH—J. S. BACH:** Brandenburg Con. No. 3; Jesu, Joy of Man's Desiring; etc. (11-68)—*Carlos, MOOG*
Col. **MS-7286;**▲18-11-0144;●16-11-0144—**CARLOS:** Well-tempered Synthesizer
Col. **MS-7355—STOCKHAUSEN:** Mikrophonie I for Tamtam, 2 Microphones, 2 Filters & Potentiometers (1964)—*Aloys Kontarsky, Alings, Fritsch, Bojé, Stockhausen, Davies, Spek;* Mikrophonie II for Choir, Hammond Organ & Ring Modulators (1965)—*Alfons Kontarsky, Fritsch, Schernus*
Col. **M-30383;**▲MA-30383;●MT-30383—**KAZDIN & SHEPARD:** Everything You Always Wanted to Hear On the Moog (9-71)
Col. **M-30683; Q—SUBOTNICK:** Sidewinder—Subotnick
2-Col. **PG-31234;**▲GA-31234;●GT-31234—**CARLOS:** Sonic Seasonings (1972)
Col. KC-31480—**CARLOS:** Clockwork Orange
Col. **XM-32088;** ▲XMA-32088; ●XMT-32088—**CARLOS:** By Request
Col. **KM-32659;** ▲KMA-32659; ●KMT-32659—**Bach:** Suite No. 2 In b; Sheep May Safely Graze; Brandenburg Con. No. 5; etc.—*Carlos, MOOG*
Col **M-32741; Q—SUBOTNICK: 4 Butterflies**
Corn. U. 1—*See Bielawa*
Corn. U. 7—*See Borden*
CRI **S-204**—*See Davidovsky*
CRI **S-219**—*See Luening*
CRI **S-227—LUENING & USSACHEVSKY:** Concerted Piece for Tape Recorder & Orchestra—*Col.-Princeton Electronic Music Ctr.*; **POWELL:** Events, M. (1963); Improvisation (1963); Second Electronic Setting (1962); 2 Prayer Settings—*Wilson, Schwartz, Davenny; Bressler, Kaplan, Tarack, Lynch, Kouguell;* **USSACHEVSKY:** Of Wood & Brass; Wireless Fantasy—*Col.-Princeton Elec. Music Ctr.*
CRI **S-255**—*See Druckman*
2-CRI **268—Varèse:** Déserts (1954-61); **LUENING:** In the Beginning (1956); **BABBITT;** Vision & Prayer (1961); **USSACHEVSKY:** Computer Piece No. 1 (1968); 2 Sketches (1971); **SMILEY:** kalyosa (1970); **SHIELDS:** Transformation of Ani (1970); **DAVIDOVSKY:** Synchronisms No. 5 (1969)—*Col.-Princeton Elec. Music Ctr.*
CRI **S-296—EATON:** Mass—*Hirayama, White, Syn.;* Blind Man's Cry—*Hirayama, Syn.;* Concert Music for Solo Clarinet—*Smith*
CRI **S-310—HILLER & BAKER:** Computer Cantata—*U. of Ill. Contemp Ch. Players, Hamm,* **MELBY:** 91 Plus 5—*Contemp. Brass Qn*
CRI **S-328—RANDALL:** Music for the Film "Eakins"—*Princeton U. Comp. Ctr;* **CEELY:** Elegia—*RAI Studio Fonologia; Mitsyn Music ; BEEP;* **DEL MONACO:** Electronic Study No. 2 (1970); Metagrama—*Sanoja, Col.-Princeton Music Ctr*
CRI **S-335—P. McLEAN:** Dance of Dawn (1974);**B. McLEAN: Spirals** (1973)—*Indiana U. South Bend Elect. Music Studio*
CRI **S-348—DODGE:** In Celebration; Speech Songs; The Story of Our Lives
Deram **18066—SATIE:** Electronic Spirit of Erik **Satie—Moog** Synthesizer,*Camarata, Contemp. Ch. Orch.*
Desto **6466—LUENING:** Fantasy In Space; Invention On 12 Notes; Legend; Low Speed; Lyric Scene; Moonflight; **LUENING & USSACHEVSKY:** Incantation; **USSACHEVSKY:** Sonic Contours
4-Desto **6474/7**—*See Tanenbaum*
Desto **7130;** ●47130—**TANENBAUM:** Movements; Contrasts; Blue Fantasy; For the "Bird"—ARP Synthesizer
DG **138811—STOCKHAUSEN:** Gesang der Jünglinge (1955/6):Kontakte (1959-60)
2-DG **2707039—STOCKHAUSEN:** Hymnen: Anthems for Electronic & Concrete Sounds
Ev. **3132—CAGE:** Variations IV (excerpts) (1965)
Ev. **3230—CAGE:** Var. IV, Vol. 2—*Cage, Tudor*
Fin. **9001—MIMAROGLU:** Wings of the Delirious Demon & other electronic works
Fin. **9002—DUBUFFET:** Musical Experiences
Fin. **9003—MIMAROGLU:** Music for Jean Dubuffet's Coucou Bazaar
Fin. **9005(Q); Q▲—SALZMAN:** Helix—Nosp*QUOG Music/Theatre Ens.;* Wiretap—*Nagrin;* Larynx Music—*Ross, Silverman;* Queens Collage
Fin. **9010(Q); ▲—BABBITT:** Ensembles for Synthesizer; **SMILEY:** Eclipse, **SHIELDS:** Farewell to a Hill; **USSACHEVSKY:** Piece for Tape Recorder, **DAVIDOVSKY:** Electronic Study No. 3, **AREL:** Stereo Electronic Music No. 2
Folk. **6301—HIGHLIGHTS OF VORTEX—JACOBS:** Chan (1956), Electronic Kabuki Mambo (1955); Logos (1956), Rhythm Study #8 (1957), **LONGFELLOW:** Notes On the History Of the World (1959?); 350-2 (1959?); **LOUGHBOROUGH:** For the Big Horn (1957); **TALCOTT:** Loop Number 3 (1957), Trilogy (1957)
Folk **33436—ELECTRONIC MUSIC—GRAUER:** Inferno, **IVEY:** Pinball (1965), **ROBB:** Collage (1964); **LeCAINE:** Dripsody (1955), **AITKIN:WALTER-OLNICK-SCHAEFFER:** Summer Idyl Noesis (1962); **M. SCHAEFFER:** Dance R 43 (1961), **STEPHEN:** Fireworks, Orgasmic Opus
Folk, **33435—ROBB:** Rhythmania & Other Electronic Musical Compositions
Folk, **33437—APPLETON:** World Music Theater
Folk **33438—ROBB:** From Razor Blades to Moog
Folk, **33440—NELHYBEL:** Outer Space Music
Folk. **33441—MIMAROGLU:** Tract (1972-74)—*Tuly Sand, American Center For Students & Artists (Paris), Columbia-Princeton Electronic Music Center*
Folk **33869—COPE:** Navajo Dedications
Folk. **33904—AIN:** Used To Call Me Sadness—*Matsua;* **CHADABE:** Echoes—*Williams,* **McMILLAN:** Whale I, Carrefours; **MUMMA:** Cybersonic Cantilevers, **USSACHEVSKY:** Conflict
Fin **9010(Q); Q▲—BABBIT:** Ensembles for Synthesizer, **SMILEY:** Eclipse, **SHIELDS:** Farewell to a Hill, **USSACHEVSKY:** Piece for Tape Recorder, **DAVIDOVSKY:** Electronic Study No 3, **AREL:** Stereo Electronic Music No. 2
GC **S-4085**—*See MacInnis, Trythall*
GC **S-4092—KNIGHT:** After Guernica (1969), Refractions for Clarinet & Tape (1962)—*Sweetkind,* Origin of Prophecy (1964); Luminescences (1967)
Lyr. **7210—SAHL:** Tropes on the Salve Regina
Main. **5002—BRYANT/CURRAN/RZEWSKI/TEITELBAUM/VANDOR :** Live Electronic Music Improvised—MEV Rome, **CARDEW/GARE/HOBBS/PREVOST/ROWE:** Live Electronic Music Improvised—AMM London
Main **5003**—*See Kagel*
Main **5004**—*See Berio*
Main. **5005**—*See Cage*
Main **5010—ASHLEY:** Purposeful Lady Slow Afternoon, **BEHRMAN:** Runthrough, **LUCIER:** Vespers (1968); **MUMMA:** Hornpipe (1967)—Sonic Arts Union
Main **5015**—*See Cage*
MCA **2220—SANTO:** John Santo Plays Bach
Mer **80000;** ▲8-80000, ●4-80000—**GLEESON:** Beyond the Sun (Holst Planets)
None. **71174—SUBOTNICK:** Silver Apples
None **71198—RUDIN:** Tragoedia
None **71199—GABURO:** Antiphonie III (Pearl-White Moments), Antiphonie IV (Poised), Exit Music I·The Wasting of Lucrecetzia, Exit Music II·Fat Millie's Lament—*New Music Choral Ens , Univ of Ill Contemp. Ch Players*
None **71208—SUBOTNICK:** Wild Bull
None **71223—ERB:** Reconnaissance (1967)—*Erb, Douglas, Forbes, Grierson, Watson, Thomas, Stein;* In No Strange Land (1968)—*Dempster, B Turetzky*
None **71224**—*See Cage & Hiller*
None **71225—WUORINEN:** Time's Encomium, for synthesizer (1968-9)
None **71245—RANDALL:** Quartets in Pairs (1964); Quartersines (1969), Mudgett (monologues by a mass murderer) (1965)—*Kessler;* **DODGE:** Changes; **VERCOE:** Synthesism—*Columbia-Princeton U's Computer Centers*
None. **71246—XENAKIS:** Bohor I (1962); Orient-Occident III (1959-60), Diamorphoses II (1958); Concret P-H II (1958)—*ORTF, Paris, Groupe recherches musicales*
None **71250—DODGE:** Earth's Magnetic Field—*Boller, Frederick, Ungar, Columbia U. Computer Center*
None. **71289—KORTE:** remembrances, for Flute & Tape;**DAVIDOVSKY:** Synchronisms No. 1, for Flute & Tape; **KUPFERMAN:** Superflute, for Flute & Tape—*Baron*
2-None. **HC-73018**($11.88, w. book)—**NONESUCH GUIDE TO ELECTRONIC MUSIC**
Odys **32160156**—*See Cage, Ichiyanagi, Lucier*
Odys. **32160158**—*See Mumma*
Odys **32160160—MAXFIELD:** Night Music (1960); **OLIVEROS:** I of IV (1966), **REICH:** Come Out
Odys **Y-34139—CANN:** Bonnylee (1972); **GRESSEL: Points In Time (1974) KREIGER:** Short Piece (1974); **LANSKY:** mild und Leise (1973-1974), **SEMEGEN:** Electronic Composition No. 1 (1971-1972); **WRIGHT:** ElectronicCompostion No. 2 (1973), **ZUR:** Chants, for Magnetic Tape (1974)
Odys **Y-34158—SUBOTNICK:** Until Spring
Op One 7—*See Schubel*
Op One 17—**McCLELLAN;** Distant Voices (1971), **CHADABE;** Ideas of Movement at Bolton Landing (1971)
Opus One**24—McCLELLAN:** Genesis (1974); Interruptions (1971)
Orion **7021;**●CAS-70211—**SWICKARD:** Sermons of Saint Francis, for narrator and tape (Moog) (1968), Hymn of Creation for narrator and type (1969)—*DuBay;* **HELLER:** Labyrinth for Cello & Tape (1969)—*Ischar*
Orion **74142—GRAYSON:** Live Electronic Music
Orion **75192—B. McLean:** II, for Piano & Tape—*Hamilton,* The Sorcerer Revisited; Genesis
Point 101—*See Badings*
RCA **ARL1-0488; Q;** ▲ARS1-0488; **Q▲**, ●ARK1-0488—**TOMITA:** Snowflakes Are Dancing (Music of Debussy)
RCA **ARL1-0838; Q;**▲ARS1-0838, **Q▲**, ●ARK1-0838—**TOMITA:** Pictures at an Exhibition (Mussorgsky)
RCA **ARL1-1312; Q;** ▲ARS1-1312; ▲; ●ARK1-1312—**TOMITA:** Firebird (Music of Stravinsky, Debussy, Mussorgsky)
Ser **12045—CUSTER:** Found Objects No. 5 for 5 Instr. & Tape—*Custer, Ens.,* No. 8, for Violin & Tape—*Kobialka;* No 6, for Flute & Tape—*Shansky;* No. 3, for Contrabass & Tape—*Turetzky*
Van **C-10057—RANDALL:** Lyric Variations for Violin & Computer (1966-7)—*Zukofsky;* **SAHL:** Mitzvah For the Dead, for Violin & Tape (1966-7)—*Zukofsky*
Van **C-10069—CZAJKOWSKI:** People the Sky
Varese **81001—TAYLOR:** Lumière, for Synthesized & Concrete Sound
West. **8110;**▲F8110—**GASSMANN:** Electronics: Music to the Ballet (1961). **SALA:** Five Improvisations
West. **8129;**▲F8129—**MEYERS:** Rhythmus; Excitement, In Memoriam for Soprano & Tape; Chez dentiste; Moonflight Sound Pictures—*Hansel,* Intervals I, **MARDIROSIAN:** Fantasia for Organ & Tape—*Mardirosian,* **HEINTZ:** Fanfare & Raga for Bassoon & Tape—*Heintz; Meyers, Catholic University of America Electronic Music Laboratory*
West. **8182—Unusual Classical Synthesizer**—*Hankinson, Putney V C.S 3 Synthesizer*

We Challenge Comparison

New $18 Gram-o-phone

WITH A $350 OR ANY OTHER PRICED TALKING MACHINE

Gram-o-phone to use its regular 16-inch horn

Other machines may use their 96-inch horns or larger

Gram-o-phone to use its 50c. records—Flat, signed, indestructible

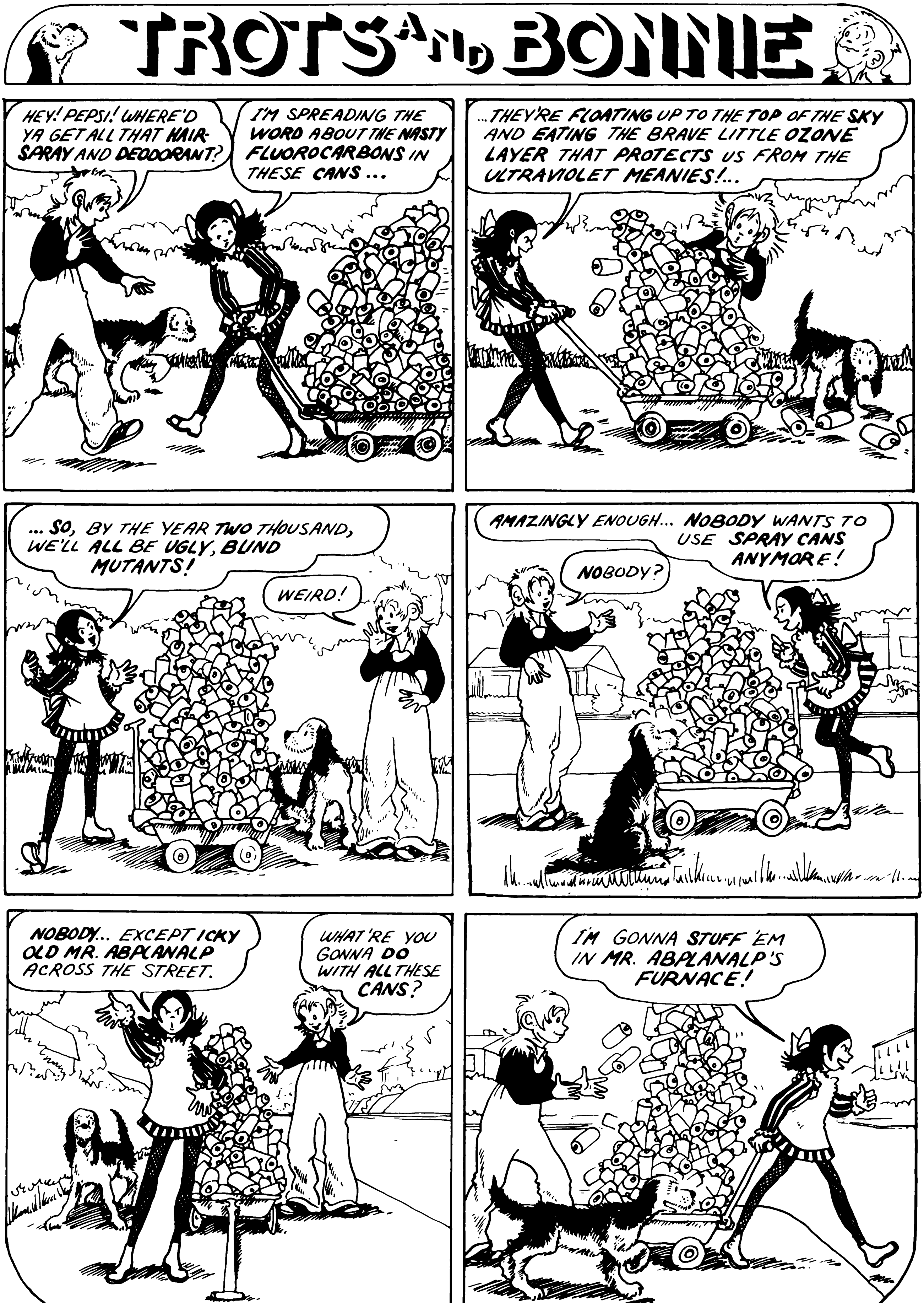
Other machines may use their 5.00 records—cylindrical, waxy, faked" or otherwise

1899

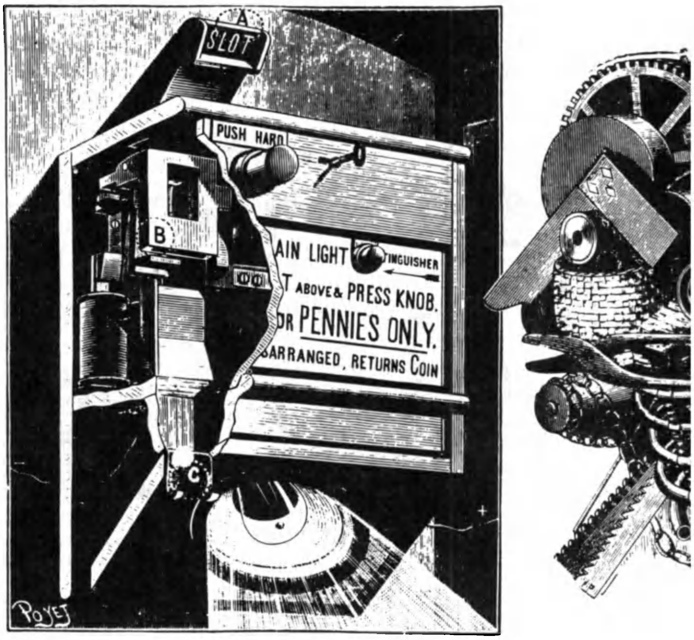
WE MAKE NO TOYS

AND LET THE PUBLIC JUDGE

National Gram-o-phone Cor., 18th STREET AND BROADWAY, NEW YORK


TROTS AND BONNIE
HEY! PEPSI! WHERE'D YA GET ALL THAT HAIR-SPRAY AND DEODORANT?
I'M SPREADING THE WORD ABOUT THE NASTY FLUOROCARBONS IN THESE CANS...
...THEY'RE FLOATING UP TO THE TOP OF THE SKY AND EATING THE BRAVE LITTLE OZONE LAYER THAT PROTECTS US FROM THE ULTRAVIOLET MEANIES!...
... SO, BY THE YEAR TWO THOUSAND, WE'LL ALL BE UGLY, BLIND MUTANTS!
WEIRD!
AMAZINGLY ENOUGH... NOBODY WANTS TO USE SPRAY CANS ANYMORE!
NOBODY?
NOBODY... EXCEPT ICKY OLD MR. ABPLANALP ACROSS THE STREET.
WHAT'RE YOU GONNA DO WITH ALL THESE CANS?
I'M GONNA STUFF 'EM IN MR. ABPLANALP'S FURNACE!
©75 SHARY FLENNIKEN

Computer His



1. The use of algorithms is fundamental to developing computer programs. When was the first known use of the floating-point algorithm, which included conditional branches and iterations?
a. Before 1940. b. 1940-1945. c. 1945-1948. d. After 1948.

2. Before the digital computer age, astronomers had to calculate orbits of astronomical bodies by hand. What was the typical precision of these orbital calculations?
a. 2 places. b. 6 places. c. 9 places. d. 100 places.

3. The government funded development of a computing machine, but cost overruns of more than ten times the original estimates caused the administration to withdraw in alarm. An oft-repeated story, but who was the developer the first time it happened?
a. W. Burroughs. b. N. Dodge. c. C. Babbage. d. G. Boole.

4. The first wholly key-operated calculating machine was the Comptometer, a practical adding and listing machine. Who invented it?
a. Leo Evans. b. Norbert Dodge. c. Francis Galton. d. Dorr Felt.

5. 'The American Arithmometer Company patented the first practical recording adding machine in 1892. Today the firm is known by another name. What is it?
a. Burroughs Corp. b. Honeywell, Inc. c. Sperry Rand Corp. d. NCR Corp.

6. Herman Hollerith organized the Tabulating Machine Corporation which grew into IBM. His punched cards were first used by which government agency?
a. Federal Bureau of Investigation. b. Census Bureau. c. Internal Revenue Service. d. Department of Defense.

7. Leonardo Torres y Quevedo's "telekino" was an early attempt to program a machine to play a popular game. What was the game?
a. Chess. b. Checkers. c. Tennis. d. Bowling.

8. The application of computers to astronomy was revolutionary. Ephemeris data calculated by Wallace Eckert in 1951 is still in use today and was the basis for many NASA missions. Which of these bodies was in Eckert's data?
a. Jupiter. b. Mercury. c. Earth. d. Bardot.

9. The "Universal Computer" designed by Alan Turing can solve any mathematical problem.
a. True. b. False.

10. Weather predicting may sometimes be no more accurate today than when granny's aching bones signalled a storm. The first person to use data processing in meteorological prognostication got his name attached to a basic weatherman's parameter. Who was he?
a. Zucker Kurtz. b. Lewis Richardson. c. Lenwood Walters. d. Charles Precipitation.

11. Plankalkul was an attempt in 1940 to?
a. Make tires from garbage. b. Develop a new memory device for computers. c. Write computer programs. d. Invent a breakfast pastry.

12. Vannevar Bush, the man who coinvented a differential analyzer with Samuel Caldwell also?
a. Discovered a cure for polio. b. Was the military R&D Director during the Manhattan Project. c. Developed the transistor. d. Sold botanical specimens door-to-door.

13. Cybernetics, which is used in control theory, automation, and computer programming was a word coined by?
a. B. Masterson. b. S. Curuthers. c. S. Morse. d. N. Wiener.

14. It was 51 feet long and 8 feet high, tipped the scales at a dainty 35 tons, and used 3,000,000 connections for its 500 miles of wire. Its name?
a. Mark I. b. King Kong. c. ENIAC. d. Jaws.

15. A government-financed machine was developed in conjunction with the Morse School of Electrical Engineering at the University of Pennsylvania. The coinventers were?

ry Trivia Quiz

Edward Pasahow

a. Martin & Lewis. b. Mauchly & Eckert. c. Wells & Fargo. d. Aiken & Wiener.

16. In the genealogy of data processing machinery, the younger brother (sister?) of BINAC was?
a. UNIVAC I: b. IBM 701. c. EDSAC. d. GENIAC.

17. UNIVAC stands for?
a. Universal Alternating Current. b. University of Alta California. c. Unit Versatile Automatic Computer. c. Universal Automatic Computer

18. The conclusion of the judge in the patent infringement suit between Sperry Rand and IBM in 1973 was that the inventor of the computer was?
a. John Atansaoff. b. John Eckert and John Mauchly. c. All of the above. d. None of the above.

19. Which university gave up the fulfillment to many future fund drives when it failed to obtain patents on early computer work done there?
a. Slippery Rock. b. MIT. c. Washington State. d. Iowa State.

20. With its 40 racks of equipment and 20,000 vacuum tubes, ENIAC ran up daily electric bills of?
a. 1¢ (during the war, electricity was free). b. $30. c. $60. d. $100,000.

21 Jay W. Forrester, an electrical engineer and management expert who applied computer simulation to the real world, also?
a. Devised COBOL. b. Retired on the proceeds at age 23. c. Founded the RAND Corporation. d. Invented the random-access memory cores.

22. The stored-program machine, program labels and symbolic addresses, macros, and microprogramming all received early attention of Maurice V. Wilkes at which lab?
a. Oxford University. b. U.S. Army Abeerdeen Proving Ground. c. Dr. Faustus. d. University of Cambridge.

23. The earliest method of storing programs in a computer used tanks containing?
a. Mercury. b. Water. c. Chicken soup. d. Liquid hydrogen.

24. Dr. Grace Hopper was involved with which commercial effort?
a. IBM 701. b. Business programs for the Burroughs 7B. c. UNIVAC I. d. HAL.

25. The first programming language was developed in 1952 for UNIVAC I and was used for numeric and scientific applications. It was?
a. FORTRAN. b. Short Code. c. Macro Language. d. Fuzz.

26. The first commercial, large-scale binary computer used the well known (by ancient, moss-covered programmers) Speed Code. The language made the single-address, fixed-point computer appear to be a three-address, decimal, floating-point computer with index registers. The computer was?
a. IBM 701. b. UNIVAC II c. Burroughs 102. d. CDC 33.

27. Arguments on optimal storage structures have been going on since the early 50s. Match the machine with its storage technique.
a. UNIVAC I. b. IBM 702. c. Blocked records on tape. d. Unblocked records on tape.

28. The man who thought of "THINK" was?
a. A.A. Michelson. b. W.S. Burroughs. c. T.J. Watson d. S. Craig.

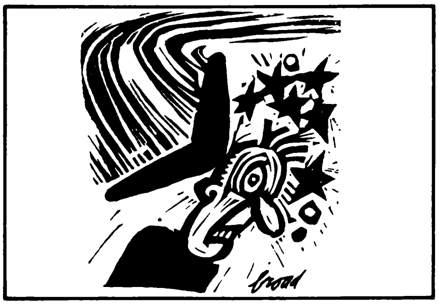
29. Wassily W. Leontief was awarded the Nobel Prize for his computer-associated input-output analysis. His work addressed?
a. Computer communications with peripheral equipment. b. Grocery store POS terminals. c. Economics. d. National defense.

30. In the top 50 data-processing industrial companies, one is a giant. IBM's share of last year's total DP revenue pot was about?
a. 25%. b. 50%. c. 60% d. 75%

answers are on page 169

Does This Question Apply to You?

Adam Yarmolinsky, loyal Harvard alum, class of '43, and now the Ralph Waldo Emerson Professor at the University of Massachusetts, received one too many thirtieth-reunion questionnaires. ("What's your income bracket?" "Are you a Republican, Democrat, or other?" etc.) Following the ancient political maxim "you can't beat something with nothing," he offered in the *Harvard Bulletin* the following "counter-questionnaire" to his classmates and others:



How often are you astonished these days?
By yourself?
By others?

Do you laugh much?
About what?

Do you cry ever?
About what?

When did you last forget yourself?
Under what circumstances?

What is the most interesting thing about your life?
The most boring?

If you had it to do over again, would you be a different person?
In what ways?

Do your children understand you?
What do you hope for your children?
What do you fear?
What do you expect?

If you had to be someone else, who would you be?

Are you more likely to be the victim or the perpetrator?

Does God love you?
Do you care?

Where are you likely to spend eternity?

What do you think about when you wake up at three in the morning?

Why do you drink so much?
Why doesn't this question apply to you?

What's most wrong with this world?
Is it anybody's fault?
Whose?
Can it be fixed?
Is it likely to be?
What, if anything, are you likely to do about it?

As between polygamy and polyandry, which would you prefer?

Does space exploration excite you?
Bore you?

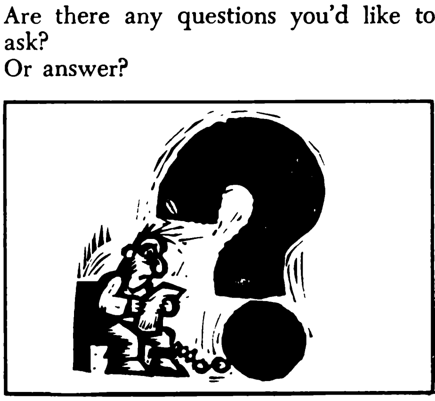
How do you feel about your dreams?
How do the people in your dreams feel about you?

What has been your greatest disappointment?

Whom are you trying to impress?
Whom are you trying to show up?

What are you really curious about?

Which of these questions was the hardest to answer?
Which was the easiest?
Are there any questions you'd like to ask?
Or answer?



The Future of Computing:

What Do YOU Think?

We would like you to help us explore the future of computing as part of a study for *Creative Computing* magazine. The questionnaire below contains three sections. The first asks for some basic background information about you, your contact with computers, and your general views about the future. The second section contains statements about developments related to computing which may or may not occur in the next twenty-five years.

We would like you to rate each item in terms of how likely you feel its occurrence will be, how important it will be if it occurs, and how desirable it will be if it occurs. These hypothetical results were generated in part by the *Creative Computing* readership and in part by reference to previous studies of the future of computing. Some statements may seem overly vague, others too specific. Please try to evaluate each in terms of the general trend it represents. The final section of the questionnaire asks for your ideas, comments and suggestions on the future of computing. Please feel encouraged to elaborate on them.

When you've completed the questionnaire, please mail the results to:

Craig Johnson
The DaVinci Group
235 Oak Drive
Willowdale Lake
North Canton, Ohio 44720

QUESTIONNAIRE

Section I

(1) ____Age

(2) Sex
____1. Female
____2. Male

(3) Education
____1. Less than high school graduate
____2. High-school graduate
____3. Some college
____4. Bachelor's degree
____5. Master's degree
____6. Doctoral degree

(4) How much background or training in computer science do you have?
____1. No background
____2. Slight background
____3. Significant background (some formal training)
____4. Very substantial background (professional employment or professional degree in computer science)

(5) About how often have you used computers directly in the last year?
____1. Rarely (a few times a year or less)
____2. Occasionally (once a month)
____3. Frequently (once a week)
____4. Daily

(6) On a scale from 1 to 5, how would you describe your political beliefs?
____1. Very conservative
____2. Conservative
____3. Middle of the road
____4. Liberal
____5. Very liberal

(7) How do you feel about your *personal* future?
____1. Very pessimistic
____2. Somewhat pessimistic
____3. Neutral, can't say
____4. Optimistic
____5. Very optimistic

(8) How do you feel about the future of the *United States?*
____1. Very pessimistic
____2. Pessimistic
____3. Neutral, can't say
____4. Optimistic
____5. Very optimistic

(9) How do you feel about the future of the *world?*
____1. Very pessimistic
____2. Pessimistic
____3. Neutral, can't say
____4. Optimistic
____5. Very optimistic

(10) Do you think forecasting studies are useful?
____1. Useless
____2. Of some use
____3. Very useful

Section II

This section lists a number of events which may or may not occur in the next twenty-five years. Please rate each event in three ways, on a scale of 1 to 5:

How *likely* is it?
1. Very unlikely, almost impossible
2.
3.
4.
5. Very likely, almost certain

How *important* will it be if it occurs?
1. Very unimportant, trivial
2.
3.
4.
5. Very important, crucial

How *desirable* will it be if it occurs?
1. Very undesirable
2.
3.
4.
5. Very desirable

(1) Audio (spoken) communication will be a common input/output mode.
____Likelihood ____Importance ____Desirability

(2) Most governmental decisions will be made by debate and opinion analysis over a computer network.
____Likelihood ____Importance ____Desirability

(3) Holographic (three dimensional) audio/visual output will be in common use.
____Likelihood ____Importance ____Desirability

(4) Problems of poverty, population growth and environmental decay will be largely solved for the United States.
____Likelihood ____Importance ____Desirability

(5) Costs of computing (costs per instruction executed) will decrease by a factor of 100 from current (1977) levels.
____Likelihood ____Importance ____Desirability

(6) Fear of invasion of privacy and general technical and economic problems will prevent the development of large data "nets" (networks).
____Likelihood ____Importance ____Desirability

(7) Direct two-way brain/computer (biocybernetic) links will be common.
____Likelihood ____Importance ____Desirability

(8) Intelligent, self-aware computers (capable of passing the Turing test) will exist.
____Likelihood ____Importance ____Desirability

(9) Complex interactions with computers will be carried out by a mix of audio communications and movement through a sensor field (programming by song and dance).
____Likelihood ____Importance ____Desirability

(10) Diffusion of information and authority throughout large computer-based bureaucracies will produce widespread alienation.
_____Likelihood _____Importance _____Desirability

(11) Basic concepts of computer science will be taught in elementary school.
_____Likelihood _____Importance _____Desirability

(12) Almost all financial transactions will be carried out by computer with no physical exchange of money.
_____Likelihood _____Importance _____Desirability

(13) Problems of poverty, population growth and environmental decay will be largely solved for the world.
_____Likelihood _____Importance _____Desirability

(14) Integrated teams of artists and scientists using computer communications and data-handling will be used to attack major social and environmental problems.
_____Likelihood _____Importance _____Desirability

(15) The majority of American homes will have a computer console.
_____Likelihood _____Importance _____Desirability

(16) Costs of rapid, randomly-accessible memory (cost per bit accessed) will decrease by a factor of 10,000 from current (1977) costs.
_____Likelihood _____Importance _____Desirability

(17) Rapid increase in automation will generate large numbers of unemployed workers.
_____Likelihood _____Importance _____Desirability

(18) Most art and entertainment will be generated and viewed via computer.
_____Likelihood _____Importance _____Desirability

(19) Costs of rapid, randomly-accessible memory (cost per bit accessed) will decrease by a factor of 100 from current (1977) costs.
_____Likelihood _____Importance _____Desirability

(20) A very large "computer hobbyist" industry will develop.
_____Likelihood _____Importance _____Desirability

(21) The United States will no longer dominate global economic and political affairs.
_____Likelihood _____Importance _____Desirability

(22) Pocket computers will have independent capacity equal to current "third generation" computers (IBM 360, Burroughs 6500, UNIVAC 1108, etc.)
_____Likelihood _____Importance _____Desirability

(23) Computer/communications utilities will become the largest industry (in dollar volume of transactions) in the U.S.
_____Likelihood _____Importance _____Desirability

(24) Most major household appliances will contain microcomputers to operate them in home use.
_____Likelihood _____Importance _____Desirability

(25) Pocket computers will be capable of automatically linking to computing "nets" (networks) if they are within a mile of a two-way telecommunications channel.
_____Likelihood _____Importance _____Desirability

(26) The majority of U.S. computers will be linked into a general computing/memory network.
_____Likelihood _____Importance _____Desirability

(27) Most governmental and business decisions will be made directly by computer with little human intervention.
_____Likelihood _____Importance _____Desirability

(28) Computer-based data gathering and analysis systems will provide the basis for rapid advances in the social and environmental sciences.
_____Likelihood _____Importance _____Desirability

(29) Costs of computing (cost per instruction executed) will decrease by a factor of 10,000 from current (1977) costs.
_____Likelihood _____Importance _____Desirability

(30) Pocket-sized memory units will have capacity equal to contemporary disk memories (will have capacities of approximately 1 billion bits per cubic inch).
_____Likelihood _____Importance _____Desirability

(31) Libraries with "hard copy" (books, etc.) will be largely replaced by computer-based data files.
_____Likelihood _____Importance _____Desirability

(32) Automatic translators for natural (human) languages will be in common use.
_____Likelihood _____Importance _____Desirability

(33) Computers will exist which can comprehend standard intelligence (IQ) tests and score over 200 on them.
_____Likelihood _____Importance _____Desirability

(34) Computers which "learn" from experience and are "educated" rather than programmed will be in common use.
_____Likelihood _____Importance _____Desirability

(35) Breakdowns or errors in computer-controlled systems will cause several disasters of serious proportions (resulting in hundreds of deaths or injuries).
_____Likelihood _____Importance _____Desirability

(36) Computer-based job-search procedures will reduce unemployment and under-employment.
_____Likelihood _____Importance _____Desirability

(37) Computer hardware will be largely based on biological and biochemical circuitry.
_____Likelihood _____Importance _____Desirability

(38) The health of the U.S. population will improve because of computer-based diagnostic and health-monitoring techniques.
_____Likelihood _____Importance _____Desirability

(39) The overall quality of life for the average American will be greatly improved.
_____Likelihood _____Importance _____Desirability

(40) The overall quality of life for the average human will be greatly improved.
_____Likelihood _____Importance _____Desirability

Section III

Please give us any comments, suggestions or ideas you may have related to the future of computing, the future of the U.S. or the future of the world.

The Case of the Reader Uncovered
or
The Clue From the 41 Square Boxes

Burchenal Green

Since Sherlock Holmes is able to use an "analytical machine" on other pages of this very issue, along with the wit and quickness of mind that are so characteristic of him, to solve a most difficult problem, I thought perhaps he could help figure out who you *Creative Computing* readers are. But alas, he was unavailable, except in the example he set. I know you want to know about yourself. We all have a great curiosity these days to learn who we are, technology reshapes our lifestyles so quickly. In the absence of the eminent Mr. Holmes I turned instead to professional help from Stuart Varden, Howard Spivak and David Wilder, who respectively teach courses in Computing in Education, SPSS, and Survey Methodology at Teacher's College, Columbia. They all very kindly agreed to help me find out who you, the readers of *Creative Computing*, are, probably not realizing all the trouble I was going to be, or I, how much work is involved in doing a good readership survey.

My first task was to decide what actual physical means I could use to ask the questions, and the options, each with different expected results, were enormous. But so was the range of costs. I chose to include a survey form in the November/December issue that was comprised of one page with 41 questions of boxes to fill in, and a space for comments. The magazine could't afford the cost of binding in a pre-paid self-mailer questionnaire so we had to ask the reader's cooperation in xeroxing the survey page or in ripping it out of the magazine, and in providing your own stamp and envelope. In way of some token of compensation we offered a drawing of survey forms, the first prize winner to receive a three-year subscription or renewal of *Creative Computing* and the second and third prize winners to get a one-year subscription or renewal each. We numbered each questionnaire as it was received and on January 12th used the BASIC random-number generator in the GE Timesharing System to get three numbers. The winners were Ed Langlin, Santa Barbara, California; John Rabenaldt, Odessa, Texas; and Jim Denning, San Francisco, California.

After it was decided to trust to our readers' filling out a page in the magazine, the next decision was that of figuring out what to ask. "You don't get a chance when you're doing the analysis to second-guess your subject," David Miller warned. "Make sure you think out what you need to know and that your survey instrument asks that. You can't analyse what you forgot to ask. But," he concluded, with the resignation of a professional, "almost everybody always wishes during analysis they'd included other questions." (How right he was. Are you readers ready for another survey later this year?)

"Keep the instrument simple and short," Howard Spivak advised, when I brought him handwritten pages of questions and comments. "People are busy. If you can limit your response to a checked box, maybe, just maybe, they'll fill it in. But don't ask them for a write-up on each question. They may want to answer, and even start the form, but chances are good they'll never finish it. Then you have nothing. Keep the response to checking boxes. You're already asking the readers to supply their own envelope and stamp."

What to ask readers had to be questions that would clearly elicit what they read and wanted to read. We are fortunate enough to be able to get a large quantity of excellent articles by renowned experts in every area of the field. They fascinate me as I read them, edit them, and rush them to the typesetter, thinking, "I can't wait for our readers to find out about that." That article gets on the "must run" list for the next issue. Then the day comes for the next issue to be assembled, the *final* no-time-left number-the-pages day when the some 250 pages of great material that must have gotten in, didn't. Not to mention all the "I-wish-I-had-room-for material." Amid great wailings, bemoanings and curses, favored articles move from the "must-get-in" pile, no page number on the bottom, to the "wish-there-was-room-for" pile. It isn't an easy decision to figure what's best to include as the contents of each issue. Variety is important. Information on new items of import will always get in, as well as good source material. Aside from that, letters to the editor, phone calls, remarks overhead, that say what you've liked and read, play a large part in dictating that final decision of what will be included.

When Dave pointed out that we got more letters about "Shuffling" than almost any other single article, we knew we had to run "Shuffling Revisited," the demand was so great. In each day's mail come letters requesting information on other languages which forced us not to hold PILOT and "A Taste of APL" on decision day. What we hoped the readership survey would do would be provide us with an even more clearcut picture of what you do read and want to read, something encompassing all types of articles, to help us in our decision-day crazies.

Information about the reader was also needed to plan future directions of the magazine and to answer questions of advertisers. I like the advertising we've run in the magazine. I find it extremely helpful to know what's available on the market and I thought many readers felt the same way. What gluttons we are in this day and age for information. But many advertisers, quite understandably, are curious to know who you readers are, to ascertain if you'd be interested in their products. I needed job, age, computer usage and potential buying information. In writing the questions of the survey, these points of information needed were those kept in mind.

After 600 of you fine people sent in your forms, they were coded and key-punched. Forms that dribbled in after that weren't included in the survey results.

To get the statistics I wanted from these forms I used SPSS, the Statistical Package for the Social Sciences, which was being implemented on the Columbia Teacher's College Burrough's 4700. SPSS is one of the most popular and widely used of the statistical packaged programs. It was originally developed as a batch system on IBM machines at

Stanford University around 1965. Since 1970, development has been progressing from the National Opinion Research Center at the University of Chicago. Its popularity among people whose work was founded on statistical analysis was instantaneous and I quickly discovered why. The program is simple to learn and it performs the kind of laborious statistics researchers would rather work from than figure out. Consequently, the package is now in use in over 1,000 installations in every state in the union and in over 50 foreign countries.

For the analysis I did of the *Creative Computing* Readership Survey I used only the simple frequency-display routines. The control cards to run the program required a control word in the control field found in columns 1 through 15 and the detailed instructions necessary for that task in the specification field in columns 16 through 80.

To get all the data necessary for the frequency of occurrence of each response to each question on the survey all that was necessary was a program that read:

```
RUN NAME       CREATIVE COMPUTING SURVEY
FILE NAME      GREEN
VARIABLE LIST  ID, CARD, ARTCLE, BKRREVS, FICTON, PUZZLE, LERNPR, GAMES, LETTER,
               ADS, EDITOR, CATLOG, CARTOON, NOTICE, USEGAM, BLDCMP, MICROS,
               EDUCAT, MUSIC, ART, MEDCIN, SPACE, BUSNIS, HOMECT,
               GRPHIC, ARTINT, LEARN, FICTWT, SOCIMP, CLUBS, OTHER,
               TECNIC, CONFNC, JOB, CATWRK, CATSML, CATHOM, CATOTH
               MMMICR, MMMINY, MMCRT, MMTERM, MMDISK, MMTAPR, MMVIDO, MMOT,
               REC, CAI, RESRCH, RECORD, CONTROL, WORK, CENTER, DISPLN,
               AGE, SCIFI, LIT, SPORTS, MYSTRY, HISTRY, ADVENT,
               ID2, CARD2, BYTE, ACM, COMPTMAG, CDEC, COMWORLD, CPR, DATAM,
               DOBBS, INTFACE, MATHTCH, MM, MININEWS, PCC, SCINEWS, SCIAMER,
               JOURNAL, READERS, SAVECOPY,COMPUTER, MINICOMP, MICCOMP, CHIPS,
               TERMINAL, TELE, GRATERM, COUPLER, PERIPH, PROGCAL, HANDCAL, LEAS,
               SOFTWARE, COURSE, LNAIDS, BUYBOOK, CONSULT, BUYOTH, COMMENT
PRINT BACK     YES
INPUT MEDIUM   CARD
N OF CASES     600
INPUT FORMAT   FIXED(F4.0,32F1.0,F2.0,27F1.0/F4.0,38F1.0)
MISSING VALUES ARTCLE TO ADVENT, BYTE TO COMMENT(9)
READ INPUT DATA
FREQUENCIES     GENERAL = ARTCLE TO ADVENT, BYTE TO COMMENT
```

From this the frequencies of occurrence of each response to each question on the survey were tabulated. The way for you to get the best understanding of the result of that tabulation is probably for us to reprint the questions as they appeared in the survey with the percentage recorded from boxes checked. The following represents what is read in an issue of *Creative Computing*.

1. In an issue of *Creative Computing*, the percentage of readers who read:

	Always	Mostly	Sometimes	Never
Feature Articles	43.8	46.4	9.6	0
Book Reviews	21.5	31.0	44.2	3.4
Fiction	34.3	25.5	31.4	8.8
Puzzles and Problems	35.8	31.2	28.5	4.4
Simulations and Learning Programs	37.7	37.4	22.0	2.7
Game Programs	48.8	26.0	21.3	3.7
Letters to the Editor	36.9	32.8	26.2	3.9
Editorials	37.4	38.3	21.3	2.9
Advertisements	35.8	39.2	24.2	0.8
Compleat Computer Catalogue	40.2	28.9	27.9	3.1
Cartoons, Humor	72.8	18.0	8.1	.8
Notices	36.7	36.7	25.0	1.4
2. I use the games or programs listed	4.8	20.8	53.9	20.6

Using the "always" percentage as a means of ranking the material that was read shows cartoons and humor leading the list, with game programs, feature articles, the compleat computer catalogue, simulation and learning programs, editorials, advertisements and puzzles and problems following in order of preference.

The wording of the questionnaire is vitally important to the analysis that results. The reader was given a choice in frequency of reading between 'always,' 'mostly' and 'sometimes,' guaranteeing that those who checked they 'always' read a type of article actually did, and that those types of articles would be given preferential treatment on getting into the next issue. Humor and cartoons offer a valuable perspective to examining computers in society, especially the computer in the home. This survey would dictate even more humor should get into this and other issues. I can also predict from this that *Creative Computing's* new cartoon book, ***The Colossal Computer Cartoon Book,*** advertised in the catalog in the center of this issue, will be a smash success.

The fact that the advertisements are so highly read justified my belief that we all want to know as much as we can about what's available on the market, because so many of us want to get more equipment as soon as possible.

A more realistic assessment of overall readership for type of material, is one developed by ading the percentages of 'always' and 'mostly' read, and ranking the articles in that fashion. Using this cumulative percentage as the readership criteria, the material would be ranked as follows.

Ranking of material read in Creative Computing:	**Percentage of high readership:**
Cartoons, Humor	91%
Feature Articles	90
Editorials	76
Simulations and Learning Programs	75
Advertisements	75
Game Programs	75
Notices	74
Letters to the Editor	70
Compleat Computer Catalogue	69
Puzzles and Problems	67
Fiction	60
Book Reviews	52
2. I use the games or programs listed:	26

Of equal weight in not only selecting but soliciting material for *Creative Computing* will be the response you gave to what you would be interested in reading, which is:

SPSS-Statistical Package for the Social Sciences

For anyone who needs statistical calculations SPSS is highly recommended. Included in the growing list of analytic procedures it can perform are:

Frequency Display Routines
  a. FREQUENCY
  b. CROSSTABS
Non-parametric Statistcs
  a. Spearman/Kendall rank-order correlation routine
Analysis of Variance
  a. Analysis of Variance
  b. Analysis of Covariance
  c. Multivariate Analysis of Variance
Product Moment Correlation Coefficients
  a. Correlation
  b. Partial Correlation
  c. Regression
Miscellaneous
  a. T-Test
  b. Discriminant Analysis
  c. Guttman Sealing
  d. Transformations
  e. Weighting

Information can be obtained from SPSS, Inc., Suite 1234, 111 East Wacker Drive, Chicago, Ill. 60601 (312) 861-0933.

I would be interested in more articles about:	**Very Much**	**Some**	**Not Much**	**Waste of Space**
Building a Computer	41.4	32.9	22.6	2.9
Microcomputers	49.0	37.6	11.0	2.2
Computer use in Education	33.0	37.2	24.3	5.0
Music	27.3	27.2	35.8	9.7
Art/Graphics	39.2	36.8	20.9	3.1
Medicine	20.2	35.4	36.0	8.4
Space Exploration	35.8	37.2	21.6	5.0
Business and Industry	34.5	42.4	18.6	4.3
Home Control	49.6	33.0	15.0	2.2
Computer Graphics	52.8	34.1	12.0	1.0
Artificial Intelligence	58.6	27.1	12.5	1.5
Learning Activities	37.9	40.9	17.8	3.2
Fiction	22.9	35.3	28.6	13.1
Social Implications	21.4	41.0	28.4	9.2
Computer Club Activities	19.6	36.3	34.5	8.2
2. I would like more technical articles:	38.5	39.0	19.1	2.7
3. I would like more reprints from conference talks:	11.7	37.5	38.7	10.9

To find which material readers want to see in *Creative Computing*, high readership would be ranked by an accumulation of the percentages gotten for 'very much' and 'some.' Readership interest is:

Rank	**Interest in More Articles About:**	**High Readership Percentage**
1	Microcomputers	87
2	Computer Graphics	87
3	Artificial Intelligence	86
4	Home Control	83
5	Learning Activities	79
6	Technical Articles	78
7	Business and Industry	77
8	Art/Graphics	76
9	Space Exploration	73
10	Computer use in Education	71
11	Social Implications	62
12	Fiction	58
13	Computer Club Activities	57
14	Medicine	56
15	Music	55
16	Conference Reprints	50

It is of great interest to us that topping the list is the interest of our readers in microcomputers, computer graphics, and *artificial intelligence*. We have stated that the educational field and the hobby field are converging in interests with the advent of technology that makes the microcomputer affordable, widely accessible, and daily more versatile. Those of you who read John Lee's account of the Dynabook and Richard Vuillequez's account of the microcomputer's impact into technical education, in the May/June issue, can have little doubt that a future expansion of computers into all facets of education will come about with the microcomputer, as it never could, despite its promise, in the past. Therefore there is no surprise that those people who want to about "microcomputers" want to read about "artificial intelligence," usually associated with researchers and access to large computers and computer graphics.

As the power of affordable microcomputers increases, is it any wonder that hobbyists want to learn about artificial intelligence?

I talk about educators and hobbyists, but who are our readers? Since a percentage of you had more than one job I had to use the SPSS RECODE function and allow everyone only one job classification. With that done, you described yourself on the survey as:

Job Function	**Percentage**
Faculty: College or University	13.3
Faculty: Grades K-12	7.4
Student: College orUniversity	14.1
Student: Grades K-12	11.8
Industry:	35.5
Government:	6.1
Other	8.9

The great growth in *Creative Computing*'s readership has come from people in industry who are users of small computers at work, obviously interested in developing more ways to utilize that computer and having programs to run on it. Of the people who classified themselves as in industry, 94% have a computer at work and 36% have computers at home. These are people of dual interests: wanting more information both for their business systems, and for their new home computers.

These statistics were quickly obtained by using the CROSSTABS feature of SPSS. It tabulated the results of running JOB BY CATWK, CATHM, or the job function by whether the user has a computer at work and by whether he has a computer at home.

The frequency percentage for where a computer was utilized is:

I utilize a computer at (check all that apply):	**Percentage**
Work	69
School	47
Home	27
Other	6

Percentages for use of equipment, application, and school use are as follows:

At home I have (check all that apply):	**Percentage**
Microcomputer	18
Minicomputer	4
CRT Terminal	12
Hard Copy Terminal	14
Floppy Disk	5
Digital Cassette Tape	11
Video Display	8
Other	12
I Utilize my home computer for:	
Recreation	27
CAI	3
Research	14
Record Keeping	11
Home Control	4
Work Related	16

(If an education) I use a computer:

In a Computer Center 35

In My Discipline Which Is____________

The disciplines that were listed were coded and the the percentages of utilization by the disciplines are:

Discipline	Percentage Utilized
1. English	2
2. Math	6
3. Physics	2
4. Computer Science	5
5. Psychology	1
6. Education	2
7. Electronics	1
8. Other	7

The age of you readers is interesting. The great majority are between the ages of 21 and 35. The statistics show:

Age	Percent
Under 20	19.3
21-35	58.5
36-50	16.8
50 or over	3.3

What brought a great sigh of joy to yours truly, poring over the statistics in the middle of one dark, freezing night, after yet another long day, was to see that over 95% of you keep your issues of *Creative,* that many of you went so far as to circle the words *treasure it* on your forms. Sleepiness faded. The statistics read:

I save my copy:	Percentage
a week or two	.5
a month or two	3.4
keep it (file it, save it, treasure it)	95.8

We had been telling stores that stock the magazine that they didn't have to worry about returns, none of the material in *Creative Computing* is dated, and from everything we'd heard you readers kept your issues for future reference, use, etc. But over 95% of you readers keeping your copies is somewhat better than even I had envisioned, and a great morale booster. Thanks.

Also, more than 60% of the copies are read by more than one person, with over 17% read by four or more, which more than doubles the readership of our circulation. The statistics are:

How many people read this copy of Creative?	Percentage
One	39.1
Two	27.9
Three	15.4
more?	17.5

A lot of information about your interests can be ascertained by other literature you read so two questions asked you to check books and magazines you read, to which you responded:

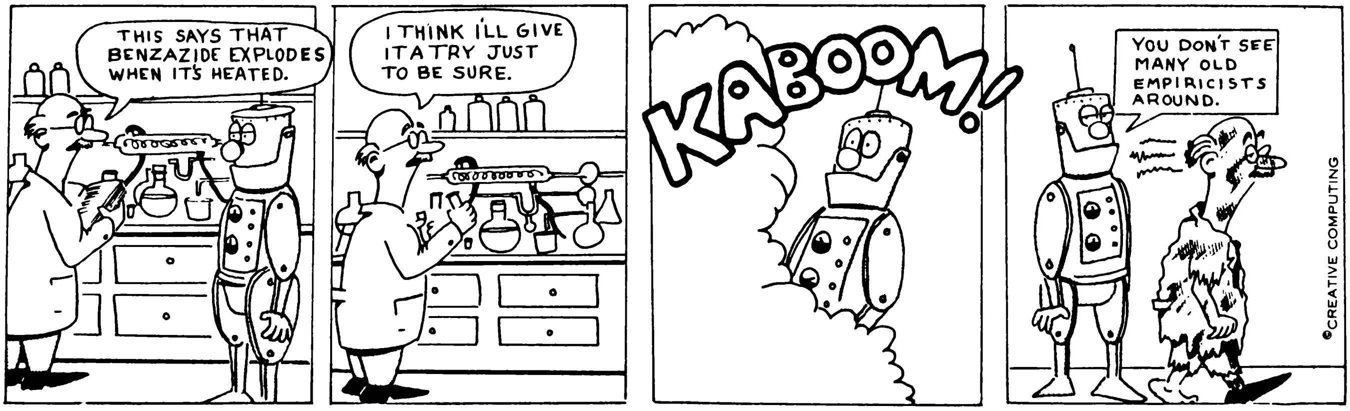
Books I read are (check all that apply):	Percentage
Science Fiction	70
Modern Literature	48
Sports	14
Mystery	32
History	35
Adventure	34

Other magazines I read are (Check all that apply):	
Byte	44
Communications of the ACM	25
Computer	13
Computer Decisions	34
Computerworld	40
Curriculum Product Review	2
Datamation	49
Dr. Dobb's Journal	12
Interface	24
The Mathematics Teacher	9
Mini Micro Systems	21
Minicomputer News	17
People's Computer Company	22
Science News	19
Scientific American	57
THE Journal	8

The highest percentage of you read science fiction over anything else listed, yet science fiction did not appear as a high priority of what you want to see more of in *Creative Computing.* Here is where some interpretation is necessary. I first ran a crosstabulation of fiction read in *Creative Computing* by want to read fiction in *Creative Computing.* Of those that like science-fiction books, 43% always read it in *Creative Computing,* 27% mostly read it, 26% sometimes read it and only 4% never read it. Of those that always read the fiction in *Creative Computing* only 53% want to see more of it very much, and of those that read it mostly only 18% want to see more of it very much. Yet over 70% of you readers are science fiction fans. I interpret. You are so anxious for more information in this rapidly growing field that although you like the stories and read them you are hesitant to have them take up space of "factual" material.

Yet although there are many good sources for science fiction, there is not enough that focus on the role the computer will have in shaping society and the interpersonal dynamics of people. It is good stuff to keep in mind as this technology advances on our life styles. It's its own kind of information. So, I think you want computer stories, but you don't want less information.

The only magazine that shares more than 50% of your readership is *Scientific American,* a magazine whose range of articles appeals to many. SPSS crosstabulations show that of those who are in industry, a field growing into the microcomputer age with dual work and personal interests,


THIS SAYS THAT BENZAZIDE EXPLODES WHEN IT'S HEATED.
I THINK I'LL GIVE IT A TRY JUST TO BE SURE.
KABOOM!
YOU DON'T SEE MANY OLD EMPIRICISTS AROUND.
©CREATIVE COMPUTING

only about 50% who read *Creative Computing* read any of the other magazines that are geared to the industrial computer user, *Computer Decisions, Computerworld, Datamation, Mini Micro Systems, Minicomputer News.* It should be of interest to an advertiser who wanted to sell to this active market of readers that by putting his ad anywhere else he would miss half of those *in industry* he reaches in *Creative Computing,* and many more in all the other disciplines.

It was most interesting to compare what job distinction signified about what equipment you buy or recommend for purchase. Although there were some job categories that listed a higher rate of selection of some items, because of a space restriction, I'll list the percentages of those in college faculty and those in industry that will buy the following items:

I recommend, specify, select or purchase: (check all that apply)	**Percentage of Faculty:**	**Percentage of Industry:**
Medium/large computer	25	27
Minicomputer	59	41
Microcomputer	46	37
MPU chips	20	18
CRT Terminal	65	51
Teleprinter	41	28
Graphics Terminal	43	25
Coupler/Data set	37	28
Peripherals	57	46
Programmable calculators	34	32
Hand calculators	38	28
Computer leasing service	20	11
Software	72	61
Courseware	37	10
Learning Aids	46	13
Books/Publications	74	48
Consultants	24	16

Creative Computing's readers have great power to buy or select an impressive array of computer equipment, a fact that is not surprising in the least, considering the mail that comes in, asking for advice and impartial reviews of equipment, but certainly indicates more equipment profile reviews are in order, and explains why the advertisements are so well read.

The survey can teach us that the computer hobbyist is not easily classified. Although 51% of the people who had microcomputers at home described their job function as industry, 11% were college faculty, 2% were high-school faculty, 5% were government workers, 11% were college students, 9% were high-school students, and 7% were in other fields.

As an example of the kind of specific information it is helpful to have and possible to get easily using SPSS, I asked for a crosstabulation of Job by Age by Micros, or job description, by age, by did you want to read more about microcomputers. For those people who wanted "very much" to read about microcomputers, the largest group was those in industry between the ages of 21-30. This was 29% of everybody who was divided into 48 groups. Of those who wanted "some" to read about microcomputers, the same group of people in industry between the ages of 21-30 had the lion's share, 31%. This group also comprised 19% of those who said that their interest was "not much" and 23% who said that such articles were "a wast of space."

This is a brief look at what can be gleaned from just who you are by the clues you left in those 41 boxes. Since I've discovered more questions I'd like to ask, next time you see a survey, fill it in. For those of you who kindly took the time, here it all is. Good likeness, eh? ■

Quiz Answers From Page 161

1. *a.* The Babylonians during Hammurabi's dynasty (1800-1600 B.C.) developed such algorithms for excavations, linear equations, and geometric problems.
2. *c.* Astronomical calculations were carried out to nine digits of precision before 1900.
3. *c.* Charles Babbage allowed the cost of his difference engine to escalate from £1,500 to £17,000 after eight years of government support.
4. *d.*
5. *a.* The Burroughs Adding Machine Co. came into being in 1905.
6. *b.* Use of punched cards cut the time required for the 1890 census to one-third that to the previous one.
7. *a.* Quevedo's machine was an early effort in the cybernetics field (even before cybernetics had a name).
8. *a.* Ephemeris data on Jupiter, Uranus, Saturn, and Pluto from 1652 to 2060 was generated.
9. *b.* Turing proved that a fixed, definite process on an automatic machine cannot solve every mathematical problem.
10. *b.* The Richardson Number relates gradients of temperature and wind velocity.
11. *c.* Karl Zuse, developer of Plankalkul, also designed and fabricated a family of computers in the 1940s, called Z1 through Z4.
12. *b.* Bush served as R&D Director during much of WWII.
13. *d.* Wiener was interested in all aspects of communications and control in living organisms and machines.
14. *a.* The Mark I, an electro-mechanical computer, was built for the U.S. Navy Board of Ordnance by Howard Aiken and his combined IBM and Harvard team.
15. *b.* ENIAC was produced for the U.S. Army Ordnance Proving Ground for ballistic calculations.
16. *a.* ENIAC begat BINAC and BINAC begat UNIVAC I.
17. *d.*
18. *c.* The judge concluded that Eckert and Mauchly derived some of their ideas from Atanasoff, partly as a result of Mauchly's visit to Iowa State in 1941.
19. *d.* See 18.
20. *c.* 20,000 vacuum tubes use a lot of watts!
21. *d.* Forrester also participated in constructing the general-purpose Whirlwind I computer.
22. *d.* Wilkes directed the mathematics laboratory at Cambridge and was the first president of the British Computer Society.
23. *a.* Mercury tanks formed the ultrasonic memories of the EDSAC machine.
24. *c.* Dr. Hopper was working for the Eckert-Mauchly Computer Corporation at the time.
25. *b.*
26. *a.*
27. *ad, bc.*
28. *c.* Thomas Watson Sr. dreamed up the slogan while working as an NCR salesman.
29. *c.*
30. *b.*

COMP SCI SERENADE

(Sung to the tune of
"My Bonny Lies Over the Ocean")

My program lies under the backlog
My card deck's all over the floor
The plotter is using a crayon
And I just can't take any more

CHORUS:
  Bring out, bring out
  Oh bring out my printout today, today
  Bring out, bring out
  The one you ripped off yesterday

The card reader chewed up my job card
And someone erased all my files
The system has been down for hours
While people collapse in the aisles

CHORUS:
  Flunk out, flunk out
  I worked like a dog each and every day
  Flunk out, flunk out
  Twelve projects were due yesterday

Security holes I've discovered
The records of grades are now mine
What once was a one-point-five average
Will soon be a three-point-nine-nine

CHORUS:
  Send out, send out
  Oh send out the grades to big companies
  Send out, send out
  They'll all want a scholar like me!

— Terry Bollinger &
The Watt Five
(Computer Science Dept.
Univ. of Missouri—Rolla)

Why I Did It

The compiler of the S-100 Computer Kit Reference List explains....

Robert Elliott Purser

In 1967, I picked up a free copy of the PDP-8 manual and have been hooked on computers ever since.

Having studied civil engineering, enology, and accounting, I went on to become a computer systems analyst, a mail-order entrepreneur on home knitting, a weekend manager of a computer store, and when times get rough—a janitor.

Recently I left my programming job with the state government to do "something useful for a change." Now I am working on all the projects I dreamed up in the last three years but never had the time to do. Besides keeping up the S-100 Computer Kit Reference List (next issue Jan. 78), I am starting another reference list. This one will catalog all software available on cassettes for Sol-20, PET, Radio Shack, and Apple II. People have asked my why I compile these lists and all I can tell them is, "It needs to be done and no one else has done it."

Since I am one of the nation's leading experts on programming designs for home knitting looms, one of the projects I am now working on is designing knits with hobby computers. In cooperation with a nationwide knitting-loom magazine, I am creating garment patterns on my computer and having them tested by knitters all across the country. My computer is an Altair 8800A, with 8K Seals, Morrow Intelligent Cassette, PolyMorphics video with keyboard, Southwest Technical printer, and most importantly, Li-Chen Wang's Palo Alto Tiny BASIC. Once the programs work they will be converted to the Sol-20, PET, and Radio Shack. Knitters will be able to create garment patterns from their own measurements and from the weight of their own yarn. (Can't you just see Granny in a rocking chair next to the fireplace, knitting patterns displayed on her computer?)

At the same time I am writing programs to emulate the computer-on-cardboard games (such as Code Name: Sector from Parker Brothers, and Electronic Battleship from Milton Bradley). Maybe with a little prodding, these game giants will realize that a $5 cassette is cheaper than a $30 game, as well as more fun.

Then my final project, if no one else has done it, is to develop a computer program by which people set type on their own home computers. A person could key in typesetting instructions on a Sol-20 (it has upper and lower case), send the cassette to a phototypesetting plant, and receive back by mail the text, ready for paste-up.

Right now, for my S-100 Reference List I use a Xerox 800 word-processing unit with a Qume proportional spacing print wheel. By using a word-processing unit I am able to cut cost and time to one-tenth that of typesetting. If I were able to typeset on my own computer, the cost would drop even further.

I am looking for sponsors for my projects. When my savings are depleted, I will be looking for employment. I want to be involved in the more interesting areas of computing, such as developing application programs for home computers or ADAM. (ADAM is the large micro computer system from Logical Machine Corp. The ADAM language is to COBOL as BASIC is to Fortran.)

Or perhaps I will start a mass-market software company. If PET or Radio Shack computers reach their expected sales of 10,000 units a month, a mass-marketing software company can sell a lot of $5 cassettes, since every computer owner will probably buy at least three program cassettes for his computer. At the same time, substantial royalties will be paid to freelance programmers. The possibilities are ENDLESS!

S-100

Bus Compatible

Computer Kits

The S-100 Bus lists computer kits which plug directly into the S-100 bus. Only kit prices are listed unless the board is only available assembled. All products and prices are based on manufacturers' and dealers' advertisements, catalogs, etc. I try to be accurate but I sometimes make mistakes. Please let me know if I do.

Manufacturers: Please send me your catalogs. How else will I know about your S-100 products?

This reference list is free and may be reproduced.

COMPUTER SYSTEMS

Byte Shop Byt-8	349.00
Computer Power & Light COMPAL-80 (assembled)	2,300.00
Cromemco Z-1 (assembled)	2,495.00
Cromemco Z-2K	595.00
Electronic Control Technology ECT-100-8080	320.00
Electronic Control Technology ECT-100-Z80	420.00
Equinox 100	699.00
Forethought Products KIMSI, connector and KIM (6502)	370.00
IMSAI 8080 Computer (chassis, power, & CPU)	699.00
IMSAI PKG-1	4,444.00
IMSAI PKG-2	9,013.00
MITS Altair 8800B	875.00
Morrow's Micro Stuff Sigma 100	250.00
PolyMorphic Systems POLY-88 System 0	525.00
PolyMorphic Systems POLY-88 System 2	735.00
PolyMorphic Systems POLY-88 System 6	1,575.00
PolyMorphic Disk System (1 disk)	3,250.00
Processor Technology SOL-PC Single Board	475.00
Processor Technology SOL-10 Terminal Computer	795.00
Processor Technology SOL-20 Terminal Computer	995.00
Processor Technology System I	1,649.00
Processor Technology System II	1,883.00
Processor Technology System III	4,237.00
Quay AI Z-80 CPU, SIO, PIO, ROM, Programmer Board	450.00
Technical Design Labs XITAN Alpha 1	769.00
Technical Design Labs XITAN Alpha 2	1,369.00
Vector Graphics Vector I	699.00
Vector Graphics Vector I without PROM/RAM	519.00
Vector Graphics Vector I without CPU	499.00
Vector Graphics Vector I without CPU, PROM/RAM	349.00
Western Data Systems DATA HANDLER (uses MOS 6502)	179.95
Western Data Systems DATA HANDLER (barebones)	79.95

SECOND OR REPLACEMENT CPU BOARD

Affordable Computer Products AZPU (uses Z-80)	249.00
Alpha Micro Systems AM-100 (16 bit)	1,495.00
CGRS 6502	?
Cromemco ZPU (uses Z-80/4 microprocessor)	295.00
IMSAI MPU-A (requires additional boards)	190.00
MRS AM6800 CK (uses 6800 MPU)	110.00
MRS AM6800 (without the 6800 MPU chip)	78.00
MRS AM6800 PC Board	30.00
R.H.S. Marketing Piggy-Back Z80-80 (assembled)	159.95
SD Sales Z-80 CPU	149.00
Technical Design Labs Z-80 (uses Z-80)	269.00

Edition 7, June 1977

I am seeking financial support for my public service projects. If you find this reference list useful, tell a patron of the arts and sciences about my work.

Robert Elliott Purser

READ/WRITE MEMORY BOARD

Advanced Microcomputer Products Logos 8K RAM	219.95
Advanced Microcomputer Products 801C 8K RAM	207.95
Advanced Microcomputer Products 32K RAM	1,150.00
Artec 32K Memory Board (8K, 250 nS)	290.00
Artec 32K Memory Board (32K, 250 nS)	1,055.00
Associated Electronics 16K Pseudo-Static	349.95
Base-2 8KS-A	98.00
Base-2 8KS-B (450 nS)	123.00
Base-2 8KS-Z	143.00
BISI CCD Board (64K)	190.00
Crestline Micro Systems (8K, low power, assembled)	179.00
Cromemco 4KZ (4K 4MHz) (Bank selectable)	195.00
Cromemco 16KZ (16K 250 nS access and cycle)	495.00
Cybercom MB6A Blue Board (8K static)	250.00
Cybercom MB7 (16K low power static)	525.00
Data Sync 16K (assembled)	298.00
Duston 8K Memory Board (bare)	29.00
Dutronics 4KLST (4K low power static)	139.00
Dutronics 8KLST (8K low power static)	285.00
E.E. & P.S. 8K (8K static)	295.00
E.E. & P.S. 16K (16K dynamic)	599.00
E.E. & P.S. 32K (32K dynamic)	895.00
Electronic Control Technology 8KM (8K 215 nS)	295.00
Electronic Control Technology 16K RAM (16K static)	555.00
Electronic Control Technology 16K RAM (with only 4K)	169.00
Electronic Control Technology 16K RAM (with only 8K)	295.00
Electronic Control Technology 16K RAM (with only 12K)	425.00
Extensys RM64-32 (32K)	895.00
Extensys RM64-48 (48K)	1,195.00
Extensys RM64-64 (64K)	1,495.00
Franklin Electric 8K Static RAM	225.00
Godbout Econoram (4K static)	99.95
Godbout Econoram II (8K)	163.84
IMSAI RAM 4A-4 (4K without sockets)	139.00
IMSAI RAM 4A-4 (4K with sockets)	159.00
IMSAI 65K (dynamic)	2,599.00
IMSAI 32K (dynamic)	749.00
IMSAI 16K (dynamic)	449.00
Kent-Moore 4K (assembled)	107.00
Microdesign MR8 (EPROM/RAM	124.95
Micromation JUMP START (4K static)	145.00
Midwest Scientific Instruments PROM/RAM Board	95.00
Mikra-D MD-2046-4 (4K static)	205.00
Mikra-D MD-2046-8 (8K static)	345.00
Mikra-D MD-2046-12 (12K static)	485.00
Mikra-D MD-2046-16 (16K static)	625.00
MiniMicroMart C-80-4K-100 (4K blank board)	39.95
MiniMicroMart C-80-4K-200 (4K blank board plus)	49.95
MiniMicroMart C-80-4K-300S (4K 2102)	79.95
MiniMicroMart C-80-4K-300LP (4K 91L02A)	99.95
MiniMicroMart C80-4K-350LP (4K 91L02C)	129.95
MiniMicroMart C80-16K-300 (16K EMM4200)	479.95
MITS 88-4MCS (4K static)	167.00
MITS 88-16MCS (16K static)	765.00
MITS 88-S4K (4K dynamic)	155.00
Morrow Intelligent Cassette (512 static)	96.00
Mountain Hardware PROROM (256)	164.00
Omni (16K static)	459.00
Omni with paging option (16K static)	468.00
Prime Radix 40K (dynamic)	1,490.00
Prime Radix 48K (dynamic)	1,580.00
Prime Radix 56K (dynamic)	1,670.00
Prime Radix 64K (dynamic)	1,750.00
Processor Technology 4KRA (4K static with sockets)	154.00
Processor Technology 8KRA (8K static with sockets)	295.00
Processor Technology 16KRA (16K static assembled)	529.00
PolyMorphic Systems MEM-8K (8K static)	300.00
R.H.S. Marketing DYNABYTE 16K (dynamic, assembled)	485.00
J-K Electronics DYNA-RAM 16 (16K)	339.00
S. D. Sales Company 4K (4K static)	89.95
Seals Electronics 8KSC-8 (8K static)	269.00
Seals Electronics 8KSC-Z (8K 250 nS)	295.00
Seals Electronics 8KSCLM (less memory chips)	124.00
Seals Electronics 16KSC-16 (16K static)	579.00
Solid State Music MB-4 (4K 91L02A)	129.95
Solid State Music MB-4 (8K 91L02A)	209.00
Solid State Music MB-4 (board only)	30.00
Solid State Music MB-6 (board only)	35.00
Solid State Music MB-6 (8K 91L02APC static)	265.00
Solid State Music MB-7 (16K static)	525.00
Technical Design Labs Z8K (4K 215 nS)	169.00
Technical Design Labs Z8K (8K 215 nS)	295.00
Technical Design Labs Z12K (12K 215 nS)	435.00
Technical Design Labs Z16K (16K 215 nS)	574.00
Technical Design Labs Z Monitor Board with 2K RAM	295.00
Vandenberg 16K RAM (dynamic)	299.00
Vector Graphics 8K RAM	265.00
Vector Graphic Reset and Go PROM/RAM	89.00
Xybek PRAMMER (256 bytes & 1702 PROMs)	189.00

PROM PROGRAMMER BOARD

Cromemco BYTESAVER for 2704 & 2708	145.00
Mountain Hardware PROROM (AMI 6834)	164.00
Quay AI Z-80 with 2708 Programmer	450.00
Szerlip Enterprises The Prom Setter (1702A and 2708)	165.00
Xebek PRAMMER for 1702 (with 1702 & RAM)	209.00

PLUG IN SOFTWARE BOARD

Computer Kits Power-Start	165.00
Cromemco Z80 Monitor Board with PROM Programmer	220.00
Godbout 8080 Software Board	189.95
Microdesign MR8 with MM2K	224.45
Micronics Better Bug Trap (assembled)	180.00
Midwest Scientific Instruments PROM/RAM Monitor	245.00
Mountain Hardware PROROM	164.00
National Multiplex Corp No. 2 SIO with monitor	140.00
Processor Technology ALS-8 (assembled)	425.00
Processor Technology ALS-8 with SIM-1	520.00
Processor Technology ALS-8 with TXT-1	520.00
Technical Design Labs Z System Monitor Board	295.00
Vector Graphics Reset and Go (2 1702A)	129.00
Vector Graphics Reset and Go (3 1702A)	159.00

VIDEO INTERFACE BOARD - BLACK & WHITE

Computer Kits INTELLITERM (characters)	395.00
Computer Graphics GDT-I (graphics and light pen)	185.00
Environmental Interface II (monitor)	245.00
Environmental Interface III (oscilloscope)	495.00
Kent-Moore alpha (assembled)	107.00
Kent-Moore graphic (assembled)	137.00
Micro GRAPHICS "THE DEALER" (graphics and characters)	249.00
MiniMicroMart C80-VBA	149.95
MiniTerm Associates MERLIN (without memory)	269.00
MiniTerm Associates MERLIN (with memory)	303.95
MiniTerm Associates MERLIN Super Dense Graphics	308.00
Polymorphics VTI/64 (graphics and characters)	210.00
Processor Technology VDM-I (characters)	199.00
Solid State Music 64x16 (graphics and characters)	179.95

VIDEO INTERFACE BOARD - COLOR

Cromemco TV DAZZLER (graphics)	215.00

TV CAMERA INTERFACE BOARD

Cromemco 88-CCC-K	195.00
Cromemco 88-CCC-K with Camera Kit 88-ACC-K	390.00
Environmental Interface I	295.00
Environmental Interface with camera	595.00

HARDWARE MULTIPLY/DIVIDE BOARD

GNAT 8006 Module (5 u-sec. process time)	225.00
GNAT 8006 Module (2.5 u-sec. process time)	275.00
North Star Computers (floating point)	359.00

CALCULATOR INTERFACE BOARD

COMPU/TIME CT 100	195.00
COMPU/TIME C 101	149.00
MiniMicroMart C80-SCI-300	99.95

SPEECH SYNTHESIZER BOARD

Ai Cybernetic Systems Model 1000	325.00
Computalker Speech Synthesizer CT-I	395.00
Logistics Synthesizer (multipurpose)	525.00

SPEECH RECOGNITION BOARD

Heuristic Speechlab	245.00
Phonics SR/8 (assembled)	550.00

JOYSTICK INTERFACE KITS

Cromemco Joystick Kit & D+7AI/O	210.00
Cromemco Dual Joystick Kits & D+7AI/O	275.00

INTERUPT BOARD

Cromemco TU-ART	195.00
El Paso Computer Group (board only)	20.00
IMSAI PIC-8 (with internal clock)	125.00
MITS 88-VI/RTC	138.00

REAL-TIME CLOCK

Comptek CL2400	98.00
COMPU/TIME CT 100	195.00
COMPU/TIME T 102	165.00
International Data Systems SMP-88	96.00
Lincoln Semiconductor Clock and Display Driver	95.00

AC POWER CONTROL

Comptek PC3216 Control Logic Interface	189.00
Comptek PC3216 & PC3202 Power Control Unit	228.50
Comptek PC3216 & 16 PC3202 16 Channel System	821.00
Comptek PC3232 Control Logic Interface	299.00
E.E. & P.S. 115V I/O	249.00
Mullen Relay/Opto Isolator Control Board	117.00

BATTERY BACK-UP BOARD

Seals Electronics BBUC·(12 amper hours)	55.00
E.E. & P.S.	55.00

MUSIC SYNTHESIZER BOARD

ALF Quad Cromatic Pitch Generator (1 channel)	111.00
ALF Quad Cromatic Pitch Generator (2 channels)	127.00
ALF Quad Cromatic Pitch Generators (3 channels)	143.00
ALF Quad Cromatic Pitch Generator (4 channels)	159.00
Cybercom SB1 Synthesizer Kit	250.00
Galaxy Systems MG-1	299.00
Logistics Synthesizer (multipurpose)	525.00
SRS Polyphonic Synthesizer SRS-320 (assembled)	175.00
SRS Polyphonic Synthesizer SRS-321 for the SRS-320	175.00

PRINTER INTERFACE BOARD

Peripheral Vision PRT-KC Printer Kit	495.00

FREQUENCY COUNTER BOARD

International Data Systems 88-UFC	149.00

IBM SELECTRIC INTERFACE BOARD

Micromation TYPEAWAY	225.00

PROTOTYPE BOARD

Advanced Microcomputer Products Universal Proto	39.95
Artec GP-100	20.00
Cromemco WWB-2K	35.00
Electronic Control Technology PB-I	22.00
E.E. & P.S. Wire Wrap	39.00
E&L Instruments Breadboarding/Interfacing Station	241.50
Electronic Control Technology PB-I	28.00
Galaxy Systems PB-I	30.00
Homestead Technology HTC-88P (QT sockets)	138.00
Homestead Technology HTC-88PF (foil pattern)	38.00
IMSAI GP-88	39.00
IMSAI BBC-5 & PI06-6 Intellegent Breadboard System	699.00
IMSAI BBC-3 & PI06-3 Intellegent Breadboard System	464.00
MiniMicroMart C-80-WW (wire wrap type)	19.95
MiniMicroMart C-80-DIP (for point to point)	18.95
MiniMicroMart C-80-BUS-WW (wire wrap)	21.95
MiniMicroMart C-80-BUS-WW-125 (with components)	27.45
MiniMicroMart C-80-DIP-BUS (for point to point)	20.95
MiniMicroMart C-80-DIP-BUS-125 (with components)	26.45
MITS 88-PPCB	45.00
MITS 88-WWB	20.00
PolyMorphics Poly I/O	55.00
Processor Technology WWB	40.00
Sargent's Dist. Co.	25.00
Seals Electronics WWC	37.50
Tarbell Electronics	28.00
Vector 8800V	19.95
Vector 8800-A	29.95
Vector 8800-B	89.00

EXTENDED BOARD

Advanced Microcomputer Products Extender	34.95
Artec EXT-100	12.00
Cromemco EXC-2	35.00
E.E. & P.S. Extender W/C	30.00
Galaxy Systems EX-I	25.00
IMSAI EXT	39.00
MiniMicroMart C-80-EXC	24.95
Mullen (with logic probe)	35.00
Processor Technology EXB	35.00
Seals Electronics EXT	29.00
Solid State Music (less connectors)	8.00
Solid State Music (w/w connector)	12.50
Suntronics EXT-I	9.95
Vector 3690-12 (assembled)	25.00

ADAPTER BOARD

MiniMicroMart C80-BA (for MOD 8/C-MOD 80 boards)	19.95
Forethought Products KIMSI (for KIM)	125.00

CARD CAGE AND/OR MOTHERBOARD

Advanced Microcomputer Products 8 slot MB w/connectors	79.95
Byte, Inc. Byt-8	229.00
Computer Data Systems Versatile CRT (assembled)	699.95
Electronic Control Technology ECT-100	100.00
Electronic Control Technology MB-20	60.00
Godbout Motherboard (10 slot)	85.00
Godbout Motherboard (18 slot)	118.00
Integrand Research Corp. 800	200.00
Integrand Research Corp. 800A	275.00
MiniMicro Mart Expander (4 slots)	10.95
MiniMicroMart Expander (9 slots)	17.95
Morrow MotherBoard	76.00
Objective Design Crate Book (plans only)	19.95
PolyMorphic P88 Chassis	235.00
TEI Model MCS-112	316.00
T&H Engineering Low Cost Buss	149.00
Vector 18 Slot Motherboard	49.00

TERMINATION BOARD

Godbout	25.00

PROM BOARD

Crea Comp M 100/16 (16K, 2116)	485.00
Crea Comp M 100/16 (with parity)	560.00
Crea Comp M 100/32 (32K, 2116)	885.00
Crea Comp M 100/32 (with parity)	990.00
Cromemco BYTESAVER (8K)	145.00
Cromemco 16KPR-K (16K, Bank selectable)	145.00
DigiComm Byteuser (uses 2708)	65.00
Digiteck PROM CARD (2K assembled without PROMS)	56.95
Electronic Control Technology 2K ROM/2K RAM	120.00
Godbout Econorom (2K)	135.00
Godbout Econorom (4K)	179.95
Godbout Econorom (8K)	269.95
IBEX 16K PROM Board	85.00
IMSAI PROM 4-4 (4K PROM)	399.00
IMSAI PROM 4-512 (1/2K PROM)	165.00
Microdesign MR8 (for 2708)	99.50
Midwest Scientific Instruments PROM/RAM Board	95.00
MiniMicroMart C80-1702-1 (all except PROMS)	49.95
MiniMicroMart C80-2708-2 (all except PROMS)	49.95
MiniMicroMart C80-256 (boot strap board, fuse link)	34.95
MITS PMC (2K)	85.00
Processor Technology 2KRO	65.00
Seals Electronics 4KROM	119.00
Solid State Music MB-3 2K (8 1702As)	105.00
Solid State Music MB-3 4K (16 1702As)	145.00
Solid State Music MB-3 (without PROMs)	65.00
Solid State Music MB-8 (2708)	85.00
Vector Graphic Reset and Go PROM/RAM	89.00
Xybek PRAMMER for 1702 (with a 1702 & RAM)	189.00

MEMORY CONTROL BOARD

IMSAI IMM ROM Control Kit	299.00
IMSAI IMM EROM Control Kit	499.00

PARALLEL INTERFACE BOARD

Advanced Microcomputer Products (3P+S compatible)	125.00
Cromemco D+7AIO (one port with seven analog ports)	145.00
Cromemco TU-ART (2 ports)	195.00
IMSAI PIO 4-1 (one port without cables)	93.00
IMSAI PIO 4-1 & PIOM (two ports without cables)	115.00
IMSAI PIO 4-1 & PIOM (three ports without cables)	137.00
IMSAI PIO 4-4 (four ports without cables)	156.00
IMSAI PIO 6-3 (three ports and bus without cables)	139.00
IMSAI PIO 6-6 (six ports and bus without cables)	169.00
IMSAI MOI (two ports & serial & tape interface)	195.00
MicroLogic M712 (one port)	69.95
MiniMicroMart C80-P I/O (two ports)	49.95
MiniMicroMart C80-P I/O with cables C80-P I/O-540	57.45
MITS 88-4PIO (one port)	105.00
MITS 88-4PIO+PP (two ports)	148.00
MITS 88-4PIO+2PP (three ports)	191.00
MITS 88-4PIO+3PP (four ports)	234.00
Morrow Intellegent Cassette with one port	102.00
PolyMorphic VTI/32 (one input port with video)	185.00
PolyMorphic VTI/64 (one input port with video)	210.00
Processor Technology 3P+S (with sockets)	149.00
Solid State Music I/O-1 (one port)	42.00
Solid State Music I/O-1 (PC board only)	25.00
Solid State Music I/O-2 (two ports)	47.50
Solid State Music I/O-2 (PC board only)	25.00
Technical Design Labs Z Monitor Board (one port)	295.00
WIZARD PSIOB (3P+S compatible)	125.00

SERIAL INTERFACE BOARD

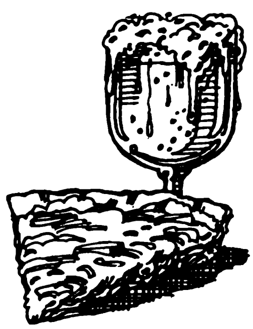
Advanced Microcomputer Products (3P+S compatible)	125.00
Cromemco TU-ART (2 ports)	195.00
IMSAI SIO 2-1 (one port, without cables)	125.00
IMSAI SIO 2-2 (two ports, without cables)	156.00
IMSAI SIO (serial, parallel, & tape interface)	195.00
Morrow Intellegent Cassette with one port	108.00
MiniMicroMart C80-SI/O-300 (TTL)	44.95
MITS 88-2SIO (one port)	150.00
MITS 88-2SIO+SP (two ports)	188.00
MITS 88 SIOB	124.00
National Multiplex Corp No. 2 SIO with ROM	140.00
Processor Technology 3P+S (with sockets)	149.00
Solid State Music I/O-2 (two ports)	47.50
Solid State Music I/O-2 (PC board only)	25.00
Technical Design Labs Z Monitor Board (two ports)	295.00
WIZARD PSIOB (3P+S compatible)	125.00

ANALOG INTERFACE BOARD

Cromemco D+7AIO (7 analog inputs & 7 outputs)	145.00
Micro Data ADC/DAC	250.00
MITS 88-ADC (assembled only)	524.00
MITS 88-Mux (assembled only)	319.00
MITS AD/DA (assembled)	235.00
PolyMorphic Systems ADA/1 (1 analog output)	145.00
PolyMorphic Systems ADA/2 (2 analog outputs)	195.00

MODEM BOARD

International Data Systems 88-MODEM	199.00
Hayes 80-103A (assembled)	279.95
Hayes 80-103A (board only)	49.95



AUDIO CASSETTE INTERFACE BOARD

Affordable Computer Products Triple Standard	135.00
DAJEN Cassette Interface	120.00
DAJEN Universal Cassette Interface (Relay Control)	135.00
IMSAI MIO (tape interface, parallel, & serial)	195.00
MiniTerm Associates MERLIN with cassette interface	298.00
MITS 88-ACR	145.00
National Multiplex Corp No. 2 SIO with ROM	140.00
Morrow Intelligent Cassette Interface	96.00
Morrow Intelligent Cassette Interface (3 drives)	102.00
PerCom Data CI-812	89.95
Processor Technology CUTS	87.00
RO-CHE with Tarbell (two ports)	215.00
RO-CHE with Tarbell (four ports)	245.00
Tarbell	120.00

TAPE DRIVE INTERFACE BOARDS

MECA ALPHA-1 System	400.00
Micro Design Model 100 (assembled)	600.00
Micro Design Model 200 (assembled)	875.00
MicroLogic M712 DG PhiDeck	69.95
National M.C. 2 SIO (R) 1 ROM	169.95
National M.C. 2 SIO (R) 2 ROM	189.95
National M.C. 2 SIO (R) with 3M3 (3M drive)	369.90
National M.C. 2 SIO (R) with 3M3 (mini 3M drive)	339.90

FLOPPY DISK INTERFACE BOARD

Alpha Micro Systems AM-200 Controller	695.00
Alpha Micro Systems AM-201 Controller	695.00
CHP Floppy Disk Controller	300.00
Computer Hobbyist Products Controller	300.00
Computer Hobbyist Products (single drive)	850.00
DigiComm 8040 Floppy Disk Controller	265.00
Digital Systems IBM compatible	1,595.00
Digital Systems dual IBM compatible	2,170.00
iCOM Microfloppy Model FD2411 (assembled)	1,095.00
IMSAI FIF	599.00
IMSAI FDC2-1 & FIF	1,694.00
IMSAI FDC2-2 & FIF	2,789.00
INFO 2000 Adapter (without RAM)	120.00
INFO 2000 Adapter (with 4K RAM)	160.00
INFO 2000 Adapter + Per Sci 1070 Controller	860.00
Micromation Universal Disc Controller	229.00
Micromation MACRO DISC System, Model 164K	900.00
Micromation MACRO DISC System, Model 256K	1,100.00
Micropolis 1053 Mod II (630K)	1,795.00
Micropolis 1043 Mod II (315K)	1,095.00
Micropolis 1053 Mod I (286K)	1,545.00
Micropolis 1043 Mod I (143K)	945.00
MITS 88-DCDD (controller & disk)	1,425.00
MITS 88-DISK	1,215.00
North Star Computers MICRO-DISK	699.00
PerCom Data Co.	695.00
Peripheral Vision interface and floppy	750.00
Peripheral Vision IFF-KC interface	245.00
Pertec RD2411	1,095.00
Processor Applications FDC-1016K Controller	395.00
Processor Technology Helios (dual)	1,895.00
Realistic Controls Z//25	1,095.00
Synetic Designs interface and floppys	2,690.00
Tarbell Bare Board Interface	40.00
Tarbell Interface	190.00

HARD DISK INTERFACE BOARD

IMSAI DISK-50	12,500.00
IMSAI DISK-80	14,700.00
IMSAI DISK-200	24,500.00
IMSAI Interface (assembled)	3,900.00

Affordable Computer Products
Byte Shop #2
3400 El Camino Real
Santa Clara, CA 95051
(408) 249-4221

Advanced Microcomputer Products
P.O. Box 17329
Irvine, CA 92713
(714) 558-8813

Ai Cybernetic Systems
P.O. Box 4691
University Park, NM 88003

ALF Products, Inc.
128 S. Taft
Lakewood, CO 80228

Alpha Micro Systems
17875 N. SkyPark North
Irvine, CA 92714
(714) 957-1404

Altair (see MITS)

Artec Electronics, Inc.
605 Old Country Road
San Carlos, CA 94070
(415) 592-2740

Associated Electronics
12444 Lambert Circle
Garden Grove, CA 92641
(714) 539-0735

Base-2, Inc.
P.O. Box 9941
Marina del Rey, CA 90291

BISI
J.R. Broom
Vancouver, B.C.

Byte Shop
1450 Koll Circle, #105
San Jose, CA 95112

CGRS Microtech, Inc.
Unknown

CHP, Inc.
P.O. Box 18113
San Jose, CA 95158

Comptek
P.O. Box 516
La Canada, CA 91011
(213) 790-7957

Computalker Consultants
P.O. Box 1951
Santa Monica, CA 90406

Computer Data Systems
English Village, Atram 3
Newark, DE 19711

Computer Kits Inc.
1044 University Avenue
Berkeley, CA 94710
(415) 845-5300

Computer Graphics Associates
56 Sicker Road
Latham, NY 12110

Computer Hobbyist Products, Inc.
P.O. Box 18113
San Jose, CA 95158
(408) 629-9108

COMPU/TIME
P.O. Box 417
Huntington Beach, CA 92648
(714) 638-2094

Computer Power & Light
12321 Ventura Blvd.
Studio City, CA 91604
(213) 760-0405

Crea Comp System, Inc.
Suite 305
4175 Veterans Highways
Ronkonkoma, NY 11779
(516) 585-1606

Crestline Micro Systems
P.O. Box 3313
Riverside, cA 92519

Cromemco
2432 Charleston Road
Mountain View, CA 94043
(415) 964-7400

Cybercom
2102A Walsh Avenue
Santa Clara, CA 95050
(408) 246-2707

DAJEN
David C. Jenkins
7214 Springleaf Court
Citrus Heights, CA 95610
(916) 723-1050

Data Sync
201 W. Mill
Santa Maria, CA 93454
(805) 963-8678

DigiComm
6205 Rose Court
Roseville, CA 95678

Digital Systems
1154 Dunsmuir Place
Livermore, CA
(415) 413-4078

Digiteck
P.O. Box 6838
Grosse Point, Michigan 48236

Duston, Forrest
885 Aster Avenue
Palatine, IL 60067

Dutronics
P.O. Box 9160
Stockton, CA 94608

E & L Instruments, Inc.
61 First Street
Derby, Conn. 06418
(203) 735-8774

E.E. & P.S.
Electronic Eng. & Production Service
Route #2
Louisville, Tennessee
(615) 984-9640

Electronic Control Technology
P.O. Box 6
Union City, NJ 07083

El Paso Computer Group
9716 Saigon Drive
El Paso, TX 79925

Environmental Interfaces
3207 Meadowbrook Blvd.
Cleveland, Ohio 44118
(216) 371-8482

Equinox Division
Parasitic Engineering
P.O. Box 6314
Albany, CA 94706
(800) 648-5311

Extensys Corp.
592 Weddell Drive, S-3
Sunnyvale, CA 94086
(408) 734-1525

Forethought Products
P.O. Box 386-A
Coburg, Oregon 97401

Franklin Electric Co.
733 Lakefield Road
Westlake Village, CA 91361
(805) 497-7755

Galaxy Systems
P.O. Box 2475
Woodland Hills, CA 91364
(213) 888-7233

GNAT Computers
8869 Balboa, Unit C
San Diego, CA 12123

Godbout Electronics
Box 2355
Oakland Airport, CA 94614

Hayes
P.O. Box 9884
Atlanta, GA 30319
(404) 231-0574

Heuristic, Inc.
900 N. San Antonio Road
Suite C-1
Los Altos, CA 94022

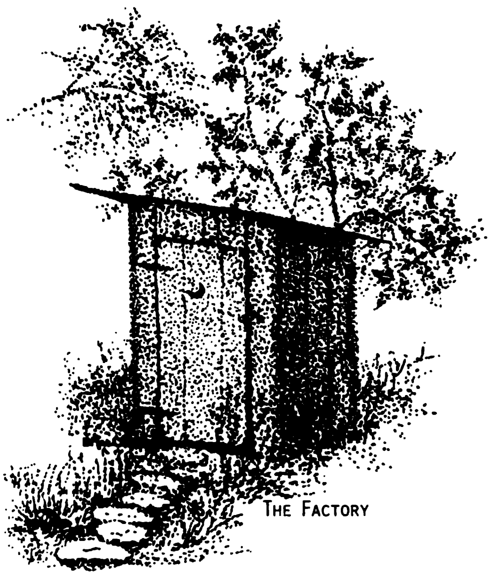
Homestead Technologies Corp.
891 Briarcliff Road N.E.
Suite B-11
Atlanta, GA 30306

iCOM Division
6741 Variel Avenue
Canoga Park, CA 91303
(213) 348-1391

IBEX
1010 Morse Avenue, #5
Sunnyvale, CA 94086
739-3770

I M S Associates, Inc.
14860 Wicks Blvd.
San Leandro, CA 94577
(415) 483-2093

INFO 2000
P.O. Box 316
Culver City, CA 90230



Integrand Research Corp.
8474 Avenue 296
Visalia, CA 93277
(209) 733-9288

International Data Systems, Inc.
400 North Washington Street,
Suite 200
Falls Church, VA 22046
(703) 536-7373

Kent-Moore Instrument Co.
P.O. Box 507
Industrial Avenue
Pioneer, Ohio 43554
(419) 737-2352

Lewis and Associates
68 Post Street, Suite 506
San Francisco, CA 94104
(415) 391-1498

Lincoln Semiconductor
P.O. Box 68
Milpitas, CA 95035
(408) 734-8020

Logistics
Box 9970
Marina Del Ray, CA 90291

North Star Computers
2465 Fourth Street
Berkeley, CA 94710

MECA
7344 Wamego Trail
Yucca Valley, CA 92284
(714) 365-7686

Micro Data
3199 Trinity Place
San Jose, CA 95124

Microdesign
8187 Havasu Circle
Buena Park, CA 90621
(714) 523-8080

Micro Designs, Inc.
499 Embarcadero
Oakland, CA 94606
(415) 465-1861

MicroGRAPHICS
P.O. Box 2189, Station A
Champaign, IL 61820

MicroLogic
P.O. Box 55484
Indianapolis, IN 46220

Micromation
524 Union Street
San Francisco, CA 94133
(415) 398-0289

Micronics, Inc.
P.O. Box 3514
Greenville, NC 27834
(919) 752-7813

Micropolis Corp.
9017 Resedo Blvd.
Northridge, CA 91324
(213) 349-2328

Midwest Scientific Instruments
220 West Cedar
Olathe, Kansas 66061

MIKRA-D, Inc.
P.O. Box 403
Hollister, Mass. 01746
(617) 881-3111

Mini Micro Mart
1618 James Street
Syracuse, NY 13203
(315) 422-4467

MiniTerm Associates
Box 268
Bedford, Mass. 01730

MITS (Altair)
2450 Alamo S.E.
Albuquerque, NM 87106
(505) 243-7821

Morrow's Micro-Stuff
Box 6194
Albany, CA 94706

MRS
P.O. Box 1220
Hawthorne, CA 90250

Mullen Computer Boards
Box 6214
Hayward, CA 94545

Mountain Hardware
Box 1133
Ben Lomond, CA 95005
(408) 336-2495

National Multiplex Corp.
3474 Rand Avenue, Box 288
South Plainfield, NJ 07080

Objective Design, Inc.
P.O. Box 20325
Tallahassee, FL 32304

Omni Systems, Inc.
P.O. Box 7536 Univ. Station
Provo, Utah 84602

PerCom Data Company
4021 Windsor
Garland, TX 75042
(214) 276-1968

Peripheral Vision
P.O. Box 6267
Denver, Colorado 80206
(303) 733-1678

Phonics, Inc.
P.O. Box 62275
Sunnyvale, CA 94086
(408) 735-7622

Polymorphic Systems
737 S. Kellogg
Goleta, CA 94608
(805) 967-2351

Prime Radix Inc.
P.O. Box 11245
Denver, Colorado 80211
(303) 433-5630

Processor Applications, Ltd.
2801 East Valley View Avenue
West Covina, CA 91792
(213) 965-8865

Processor Technology
6200-L Hollis Street
Emeryville, CA 94608
(415) 652-8080

Quay Corporation
P.O. Box 386
Freehold, NJ 07728
(201) 681-8700

Realistic Controls Corporation
3530 Warrensville Center Road
Cleveland, Ohio 44122
(216) 751-3158

R.H.S. Marketing
2233 El Camono Real
Palo Alto, CA 94306
(415) 321-6639

RO-CHE Systems
7101 Mammoth Avenue
Van Nuys, CA 91405

S. D. Sales
P.O. Box 28810
Dallas, Texas 75228

Sargent's Dist. Co.
4209 Knoxville
Lakewood, CA 90713

Scientific Research Instruments
P.O. Drawer C
Marcy, NJ 13403

Seals Electronics
Box 11651
Knoxville, TN 37919
(615) 693-8655

Smoke Signal Broadcasting
P.O. Box 2017
Hollywood, CA 90028

Solid State Music
M I K O S
419 Portofino Drive
San Carlos, CA 94070
(408) 246-2707

Stillman Research Systems (SRS)
P.O. Box 14036
Phoenix, AZ 85063

Suntronics Company
360 Merrimack Street
Lawrence, MA 01843
(617) 688-0751

Synetic Designs Company
P.O. Box 2627
Pomona, CA 91766
(714) 629-1974

Szerlip Enterprises
1414 W. 259th Street
Harbor City, CA 90710

TEI Inc.
7231 Fondren Road
Houston, Texas 77036
(713) 774-9526

T & H Engineering
P.O. Box 352
Cardiff, CA 92007
(714) 753-8568

Tarbell Electronics
20620 South Leapwood Avenue
Suite P
Carson, California 90746
(213) 538-4251

Technical Design Labs Inc.
342 Columbus Avenue
Trenton, NJ 08629
(609) 921-0321

Vandenberg Data Products
P.O. Box 2507
Santa Maria, CA 93454
(805) 937-7951

Vector Electronics Company, Inc.
12460 Gladstone Avenue
Sylmar, CA 91342
(213) 365-9661

Vector Graphic Inc.
717 Lakefield Road, Suite F
Westlake Village, CA 91361
(805) 497-0733

Western Data Systems
3650 Charles Street, No. Z
Santa Clara, CA 95050

WIZARD Engineering
8205 Ronson Road, Suite C
San Diego, CA 92111

Xybek
P.O. Box 4975
Stanford, CA 94305
(408) 296-8188

Puzzles, Problems & Programs

puzzles & problems

Thinkers' Corner

by Layman E. Allen © 1976

MATHEMATICS PUZZLES

How many of the problems (a) through (f) below can be solved by forming an expression equal to the GOAL? (Suppose that each symbol below is imprinted on a disc.)

The expression must use:

(1) only single digits combined with operators,
(2) all of the discs in the REQUIRED column,
(3) as many of the discs in PERMITTED as you wish, and
(4) at most one of the discs in RESOURCES may be used.

Special Rules

The '*' indicates "to the power of". Thus 3*2 = 3^2 = 9.

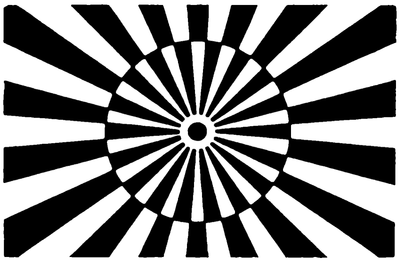
The 'V' indicates "the nth root of". Thus 3V8 = 2.

Parentheses can be inserted anywhere to indicate grouping, but never to indicate multiplication.

Problem	Goal	Required	Permitted	Resources
(a)	9	8 +	2 3 4 ÷	- ÷ V 1 3 5 8
(b)	26	4 6 x	5 x ÷	+ - x 0 1 3 5
(c)	1	2 4	1 2 ÷	+ - ÷ 6 7 8 9
(d)	7	9 -	3 5 +	- x ÷ V 2 4 6
(e)	1	4 6 ÷	1 8 +	- x ÷ V 1 3 6
(f)	9	7 4 V	4 +	+ x ÷ V 1 2 3

Some suggested answers (frequently there are others):
(a) (8 - 2) + 3
(b) (5 x 4) + 6
(c) (4 ÷ 2) - 1
(d) 3 - (5 - 9)
(e) (1 ÷ 4) + (6 ÷ 8)
(f) (2 V 4) + 7

If you enjoy this kind of puzzle, you might like playing EQUATIONS: The Game of Creative Mathematics. Free information about this and other instructional games is available upon request from The Foundation for the Enhancement of Human Intelligence, 1900-E Packard Road, Ann Arbor, MI 48104.



An Interesting Problem

by Fred Gruenberger

A financial institution advertises:

> Send us $100 per month for 12 years and we'll send you $100 per month forever.

This is simply a clever way of expressing confidence that the institution involved can maintain a 6% interest rate (compounded monthly) indefinitely. At that interest rate, $100 per month for 144 months builds up a reserve of over $21,000, and $21,000 at 6% can then generate $100 per month in interest without disturbing the principal. As a matter of fact, the build-up period could be 129 months, rather than 144, and the same offer could be made (the amount available at the end of 139 months would be $20,004.84).

They could also change the offer to read:

> Send us $100 per month for 12 years and we'll send you $200 per month for the following 12 years.

Again, the same offer could be made using 139 months instead of 144.

All the above assumes constant interest rates, and the problems could be solved quickly and readily with compound interest tables.

Suppose, however, that the interest rate were not constant. Assume that a fund of exactly $20,000 is available for payout, at an initial rate of 6%, compounded monthly, but that the interest rate rises by .00008333333 per month; that is, the rate rises to 7% at the end of ten years and continues to rise at that rate. How much can be paid back per month, so that the fund is completely depleted after 139 months?

Reprinted with permission from *Popular Computing*, Vol. 4, No. 1 (P.O. Box 272, Calabasas, CA 91302)

COMPUTER RECREATIONS

by D. Van Tassel

Syntax Messages

In the March-April 1977 issue I suggested you write a program to generate as many different syntax-error messages as possible with as few statements in the program as possible. I received a few responses but first prize must go to Wayne M. Compton of Dhahran, Saudi Arabia. He sent me a COBOL program with just one statement which generated 570 error messages. The statement was just the program name paragraph:

```
PROGRAM-ID. ERRMSG.
```

The compiler then went wild and generated 570 error messages. He ran this program on a IBM 370 OS/VS system. If you have access to such a system you might try it.

Old Time Problems

Here are several problems from two arithmetic textbooks from around the turn of the century, i.e., around 1800! One is *Schoolmaster's Assistant* by Nathan Daboll, and the other is *An Introduction to Arithmetic* by Erastus Root, 1796. Naturally, there were no computers, calculators, or slide rules in those days, yet consider the fact that these problems do not "come out even" and require a great deal of tedious hand calculation. Try them by hand and then write short programs to solve them. Which do you like best?

1. An ignorant fop wanting to purchase an elegant house, a facetious gentleman told him he had one which he would sell him on these moderate terms, viz. that he should give him a penny for the first door, 2¢ for the second, 4¢ for the third, and so on, doubling at every door, which were 36 in all. It is a bargain, cried the simpleton, and here is a dollar to bind it. Pray, what would the house have cost him? (Can you solve this problem with a 4-line BASIC program?)

2. What is the difference between six dozen dozen and half a dozen dozen?

3. Divide 4½ gallons of brandy equally among 144 soldiers.

4. How much shalloon that is 3/5 yard wide, will line 5½ yards of camblet which is 1¼ yard wide?

The Little Pigsby Farm Puzzle

The farm known as Little Pigsby has been in the possession of the Dunk Family for several centuries. One of the fields of this farm is rectangular and is known as Dog's Mead. Below are a number of clues to figures relating to the property which must be written in the appropriate places in the framework. When completed "2 down" will give the square of the age of Mrs. Gooby, Farmer Dunk's mother-in-law.

CLUES

The year of the puzzle is 1939.
4840 sq. yds. equal 1 acre.
4 roods equal 1 acre.
20 shillings equal 1 pound sterling.

1		2	3		■	4
	■	5		■	6	
	■		■	7		
■	8		9		■	■
10		■	11		12	13
	■	■	■	14		
15		■	16		■	

ACROSS

1. Area of Dog's Mead in square yards.
5. Age of Farmer Dunk's daughter Martha.
6. Difference in yds. between length and breadth of Dog's Mead.
7. No. of roods in Dog's Mead times (X) "8 down."
8. Year when Little Pigsby came into possession of the Dunks.
10. Farmer Dunk's age.
11. Year of birth of Mary, Farmer Dunk's youngest child.
14. Perimeter in yds. of Dog's Mead.
15. Cube of Farmer Dunk's walking speed in M.P.H.
16. "15 across" minus (—) "9 down."

DOWN

1. Value in shillings per acre of Dog's Mead.
3. Age of Mary.
4. Value of Dog's Mead in pounds sterling.
6. Age of Farmer Dunk's first born, Ted, who was twice as old as Mary in 1935.
7. Square of the no. of yds. in breadth of Dog's Mead.
8. No. of minutes it takes Farmer Dunk to walk one and 1/3 times around Dog's Mead.
9. See "10 down."
10. "10 across" times (X) "9 down."
12. One more than the sum of the digits in "10 down."
13. Length of tenure in years of Little Pigsby by the Dunks.

puzzles & problems

New Life for Nim!

by B. M. Rothbart, London

Nim is perhaps the most popular game as a programming exercise in beginning computer science courses. However, as far as playing the game for the initiated player, it is no more than a test of adding binary numbers in one's head.

However, a small change produces a two-dimensional Nim which resists attempts at a definite analysis. Objects are placed in small groups forming a two-dimensional array as follows:

```
2 7 6 3
1 2 0 5
4 3 9 1
```

The two players then take turns to remove matches with the sole limitation that the matches removed on any turn must be from the same column or row. As in conventional Nim, the player who takes the last match can be either the winner or loser depending on the agreement reached before the game started.

Some trivial wins and losses are readily spotted, but the charm lies in the way in which even experienced players are unable to play by rote. Is there a foolproof method for playing? We leave that for readers to find out.

Different Numbers

by
Eve R. Wirth

In this puzzle; first you must supply the 12 missing numbers and then add them up. This will give you a year in which one of the underlying principles of the computer was invented. (Hint: it was a mathematical principle.) What was the year, the principle, and who was the person?

1._____ Nights" ak/a The Arabian Nights
2._____ degrees in a circle
3._____ original colonies
4._____ signs in the Zodiac
5._____ square inches = 1 square foot
6._____ th Amendment (Women's Suffrage)
7._____ Years War (Anglo-French wars)
8._____ Heinz Varieties
9._____ Degrees (boiling point of water Fahrenheit thermometer)
10._____ feet = 1 fathom
11._____ R's (basics of education)
12._____ good turn deserves another.
_____ Total

Answer to "Different Numbers"

1.	1001	5.	144	9.	212
2.	360	6.	19	10.	6
3.	13	7.	100	11.	3
4.	12	8.	57	12.	1

The total is 1928 and if you have a good mathematics history book you will discover that Vannevar Bush devised differential analysis in that year. You will be forgiven if you could not find Bush in your book (he is not widely recognized), however, you might have found that Fleming discovered Penicillin that year and Pressey built the first teaching machine.

Simple (Crypt) Arithmetic

```
  N I N E
- F O U R
  F I V E
```

```
    O N E
    T W O
+ F I V E
E I G H T
```

```
    O N E
    T W O
+ F O U R
S E V E N
```

```
  S E V E N
  S E V E N
+     S I X
T W E N T Y
```

```
  F I V E
- F O U R
    O N E
+   O N E
    T W O
```

```
F O R T Y
    T E N
  + T E N
S I X T Y
```

MULTIPLY

```
  G E T
    O N
  R O N
G E T
G R A N
```

DIVISION

```
          M A N
B E    )A B L E
        M N
          L L
          A T
            H E
            H E
```

THE KEYRING PROBLEM

Consider a keyring with 5 keys. Because of the structure of a keyring, each key is adjacent to 2 other keys. The problem is to engrave a number on each key so that the keying possesses the following property:

For any number, n, between 1 and 21 inclusive, there exists an *adjacent* group of keys whose engraved numbers sum to n. For example: If 1-2-4-x-x is part of the keyring, here are the possibilities:

n		*keyring*
1	=	1
2	=	2
3	=	2+1
4	=	4
5	=	?
6	=	4+2
7	=	4+2+1

Clearly, it is not possible to form a 5 with adjacent keys. Can you write a program to solve this problem in a *reasonable* amount of time?

Rob Kobstad and Mike Lucey
Notre Dame, IN 46556

P.S. A Fortran solution on a GA 18/30 required less than 1 minute to compile/solve the problem.

COMPUTER RECREATIONS

by D. Van Tassel

Syntax Messages

All programmers get tired of getting syntax error messages, but there is an interesting program to write where the goal is to get syntax error messages. Now any fool can get a lot of syntax messages (just forget to declare an array) but try to get the maximum number of *different* syntax error messages. To find out how many error messages are possible check your manuals for a complete listing of syntax error messages.

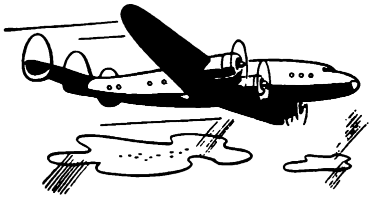
In order to not make it too easy let's try to get as many different syntax messages with as few statements as possible. We can set up a ratio as follows:

$$\text{ratio} = \frac{\text{\# of statements}}{\text{\# of different error messages}}$$

If you get a good solution send me the listing (but you must do the counting). If I get some real good solutions I will publish them in a later column. This problem is language dependent so I will try to publish solutions by language. (Send solutions to D. Van Tassel, Computer Center, Univ. of California, Santa Cruz, CA 95064).

THE FRIENDLY SKIES

Every hour on the hour a jet plane leaves New York for Los Angeles and, at the same instant, one leaves Los Angeles for New York. If each trip lasts exactly five hours, how many planes from L. A. will each plane from N. Y. see (assuming good visibility, of course)?



Thinkers' Corner

by Layman E. Allen © 1975

SET THEORY PUZZLES

How many of the problems (a) through (f) below can you solve by forming an expression that will name the number of cards in the universe that is listed as the GOAL? (Suppose that each letter and symbol below is imprinted on a disc.)

The expression must use:

(1) all of the discs in the REQUIRED column
(2) as many of the discs in PERMITTED as you wish, and
(3) exactly one of the discs in RESOURCES

Universe of Cards	1	2	3	4	5	6
	B	A B D	D	B D	C D	A C

Examples:

The expression A names 2 cards (2,6).

The expression A' (complement) names 4 cards (1,3,4,5).

The expression B∩D (intersection) names 2 cards (2,4).

The expression B∪D (union) names 5 cards (1,2,3,-4,5).

The expression B-D (difference) names 1 card (1).

Problem	GOAL	REQUIRED	PERMITTED	RESOURCES
(a)	4	∪	ACD∪	ABC∪ - '
(b)	3	B	CD∪∩	BD∪∩ - '
(c)	5	C ∪	ABD∩	BC∪∩ - -
(d)	5	C -	BC∪∩	CDD∪∩ '
(e)	3	A∩ '	BC∪∩	ABC D∩ -
(f)	4	BD -	AD∪ -	BCD∪∩ -

If you enjoy this kind of puzzle, you might like playing ON-SETS: The Game of Set Theory. Free information about this and other instructional games is available upon request from The Foundation for the Enhancement of Human Intelligence, 1900-S Packard Rd., Ann Arbor, MI 48104.

Some Suggested Answers (frequently there are others):

(a) A∪B (b) B∩(C') (c) A∪C∪B
(d) (B-C)∪D (e) A'∩D (f) (A∪D)-(B∪D)'

puzzles & problems

Number Game

Write a BASIC program to simulate this Number Game that appeared in the latest issue of *Zephrus: De-Schooler Primer*.

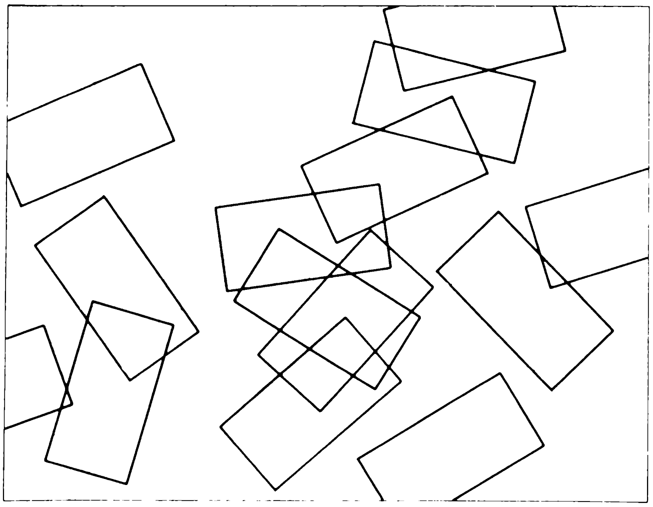
Equipment:	2 dice and a score sheet. Each die numbered from 1 to 6. Score sheet numbered from 1 to 100 with a blank beside each number.
Rules:	Each player (you vs. the Computer?) rolls the dice. The object is to fill the blanks on the score sheet using the numbers on each die in any arithmetic operation. Only one operation may be used.
Strategy:	With a roll of 2 and 6 the player could elect to fill in either the 2, the 6, the 4 (6-2), the 8 (6+2), the 12 (6×2), the 3 (6/2), the 36 (6^2), or the 64 (2^6).

The winner? The player that fills the most blanks!

Drop It

Cut out a cardboard rectangle 1-1/2" x 3". Drop it from a height of about 2 feet onto an 8-1/2" x 11" sheet of paper laid flat on a table. Outline the rectangle where it falls with a "Flair" type pen. Repeat 15 times.

Write a program to determine the probability of one rectangle touching one other, two others, and so on. How do the results from your program compare to your drawing? Try it for 30 drops. Any improvement?



False Cancellation

The equation 16/64 = 1/4 is a result obtained by the cancellation of the 6 in the numerator and denominator. Find all the cases in which AB/BC = A/C for A, B, and C integers between 1 and 9 inclusive. Do not consider obvious special cases such as 22/22, 33/33, etc.

Squared Sums

The four digit number 3025 has the following property: if the number formed by considering only the first two digits (30) is added to the number formed by considering only the last two digits (25) (the total will be 55), and if this number (55) is squared, the result will be the original number:

$$(55)^2 = 3025$$

Find all 4 digit numbers having this property. Do *not* check numbers beyond 9900 since 9901 would be arranged as

$$99 + 01 = 100 \text{ and } (100)^2 = 10000$$

which is a *5* digit number.

Sequential

What is the next number in the sequence:
9, 7, 7, 9, 13, 10, 9, ?

Too Many Coconuts

There are 3 pirates and a monkey on a desert island who have gathered a pile of coconuts to be divided the next day. During the night one pirate arises, divides the pile into 3 equal parts and finds one coconut left over, which he gives to the monkey. He then hides his share away from the pile. Later during the same night, each of the other two pirates, in turn, arise and repeat the performance of the first pirate. In the morning all 3 pirates arise, divide the pile into 3 equal shares and find one left over which is given to the monkey. How many coconuts were in the original pile? Since the result is not unique, find all values from 1 to 1000 that satisfy the conditions.

COMPUTER RECREATIONS

Dennie Van Tassel

Bust Your Compiler

Many commands within your compiler have limitations. These limits are usually so large that you will seldom encounter them. For example, one popular compiler will allow about 400 parentheses in *one* statement before objecting. An interesting exercise is to find other limitations on your favorite compiler. Here are some suggestions:

(a) Maximum number of parentheses in one statement.
(b) Maximum size of a 1-dimensional array. Maximum dimensions of an array.
(c) Maximum length of literal or bit constant.
(d) Maximum length of a single statement.
(e) Maximum length comment or maximum number of consecutive comments.
(f) Maximum number of nested DO or FOR loops (or blocks or IF THEN ELSE).
(g) Maximum number of subroutines (or nested calls to subroutines).
(h) Maximum number of arguments in a subroutine.
(i) Maximum number of recursive calls.

Can you think of any other restrictions of this type? Hint: Try examining the list of error messages for your language compiler.

Self-Reproducing Program Revisited

I received many solutions for this problem. For those of you that may have missed it in the Sep/Oct 1976 issue, here is the problem: Write a program that prints an exact copy of itself. No input statements are allowed.

Several people sent in solutions where they used the file the program was in or they created a file before hand, and then read the file. But this violated the rule that no input statements were allowed. Also there were several solutions sent in that required over a page of code.

Here are three good solutions, one in BASIC and two in FORTRAN. No COBOL solution was sent in, even though it is fairly easy in COBOL. It seems it should be possible to write a shorter BASIC version, but the solution is pretty good.

Basic solution by Donald Bell, a student at California State University at Fullerton.

```
10 DATA "B$='DATA '+CHR$(34)
20 DATA " FØR J=10 TØ 180 STEP 10
30 DATA "READ A$
40 DATA "PRINT J;B$;A$
50 DATA " IF J<>90 THEN 170
60 DATA "RESTØRE
70 DATA· "B$=''
80 DATA " NEXT J
90 DATA " END
100 B$='DATA '+CHR$(34)
110 FØR J=10 TØ 180 STEP 10
120 READ A$
130 PRINT J;B$;A$
140 IF J<>90 THEN 170
150 RESTØRE
160 B$=''
170 NEXT J
180 END
```

Fortran solution by Mark Barnett at Stanford University.

```
REAL*8F(6)/48H(7X'REAL*8F(6)/48H'6A8,1H//7X'PRINTF,F'/7X'END')/
PRINTF,F
END
```

Fortran solution by Armond O. Friend of Brookline, Mass., a Freshman at MIT.

```
     WRITE(6,100)
     CALL EXIT
100   FØRMAT(T7,12HWRITE(6,100)/T7,9HCALL EXIT/
    12(48H 100   FØRMAT(T7,12HWRITE(6,100)/T7,9HCALL EXIT/
    1/T6,6H12(48H),T69,2H)/,T7,2(31H/T6,6H12(48H),T69,2H)/,T7,2(31H)/
    1T62,11H)/T7,3HEND),T6,2(28H1T62,11H)/T7,3HEND),T6,2(28H)/T7,3HEND)
     END
```

Run Times — The Most Important Variable is the Human Factor

78 multiplied by 345 equals 26910. Notice that these three numbers have between them all of the digits 0 to 9 occurring just once. Can you write a computer program to find all such combinations?

In the Jan-Feb 1975 issue of *Creative Computing*, we posed a problem to find all of the combinations of a 2-digit number multiplied by a 3-digit number equaling a 5-digit number which used all ten integers 0 to 9. (There are nine solutions.)

Geoffrey Chase, OSB, of the Portsmouth Abbey School in Rhode Island wrote five different programs to solve the problem on the same computer (PDP 8/e) and did an exhaustive analysis of the differences. Space does not permit us to print his entire discussion or the programs; however, the following is a brief summary.

Language	*Timing*
FORTRAN/SABR Coding, using EAE subroutines	0.9 sec
FORTRAN, no machine language patches	3.2
Compiled BASIC, using EAE	15.5
FOCAL, with some EAE floating point patches	61.5
Multi-user BASIC, no EAE	108.0

We see that there is over a 100-to-1 spread with the easy-to-program languages taking considerably longer to run. However, one must ask the question whether the ultimate goal in a particular program should be efficiency in running or efficiency in coding. To give you some grist for thought, why not try to come up with an estimate of the following ratio for your computer installation.

$$\frac{\text{Cost to Program One Line of Code}}{\text{Cost to Execute One Line of Code}} = x$$

IBM estimates that the value of x for 360 and 370 series computer installations is approximately 100 million to 1. Obviously the ratio is different for a hobbyist or student programming a dedicated micro or mini. Nevertheless, the point is that the human factor is incredibly important.

That is not to say that the computer doesn't play an important role. Next issue we'll be publishing a set of timing comparison programs in Basic and Fortran along with timings on popular minis, micros and timesharing systems so you can compare your machine to others.

puzzles & problems

SOLITAIRE

Here is an interesting solitaire game which can be played on a 6x6 corner of a checkerboard or, better yet, programmed for the computer.

The first cell (top lefthand corner) is empty; all the others contain checkers, chips, beans, etc. The problem is to reduce the checkers to one and finish with the last checker in Cell 1. A move is defined as any one checker jumping one other checker in a single straight-line step (no diagonals), and then removing the jumped-over checker from the board.

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31	32	33	34	35	36

LADDER IN ALLEY

Write a program to determine the longest ladder that can be carried in a horizontal position around a corner in Genoa when a 4-meter alley meets one 2.5 meters wide.

TILE FLOORS

Two square rooms in a house were tiled with 2,120 tiles, each a foot square. Each side of one floor is 12 feet longer than each side of the other. What were the dimensions of the two floors?

COMPUTER RECREATIONS

by D. Van Tassel

Knight's and King's Tours

The knight in chess can also be used for some interesting programs. The knight can move one square vertically or horizontally and then two squares at right angles, or vice versus. Here is a program using knights: Write a program to determine the maximum number of knights which can be put on one chess board so no knight can capture another knight. If you think about this, the solution is fairly easy. I will print a solution in a later column.

A famous problem using a knight is called the Knight's Tour: Write a program with the knight starting at some position and have the knight visit each square on the chess board *once* and *only once*. There are many solutions to this problem. One solution is in the book *BASIC Programming* by Kemeny and Kurtz, John Wiley & Sons.

A final chess problem is called the King's Tour. Starting with a king in its normal position have the king visit each square once and only once.

If anyone knows of any other interesting programs to write using chess pieces, please send them to me and I will try to use them in a later column.

FLAGSTONE WALK

A man has red, gray, and blue flagstones for making a walk (one stone wide). How shall he lay them so that no pattern is immediately repeated and how many can be so laid? That is, no consecutive stones may have the same color; no consecutive pairs of stones may have the same colors in the same order; no three stones may show the same sequence as the preceding three; and so on. For purpose of computer solution, you may represent the different color flagstones by the integers 0, 1, and 2.

Thinkers' Corner

by Layman E. Allen ©

WORD PUZZLES

How many of the problems (a) through (f) below can you solve by forming a network of words that have exactly as many letters as the number listed as the GOAL? (Suppose that each symbol below is imprinted on a disc.)

To qualify as a network

(1) all sequences of discs across and down must be words,
(2) the words must have two or more letters and not be proper names,
(3) all of the discs in the REQUIRED column must be used,
(4) as many of the discs in PERMITTED as you wish may be used, and
(5) at most one of the discs in RESOURCES may be used.

Example: The number of letters in the words of the network

```
CAT    is 7:          CAT=3, TO=2, ON=2
 ON                   3 + 2 + 2 = 7
```

The number in the network CAT is 3.

Problem	GOAL	REQUIRED	PERMITTED	RESOURCES
(a)	7	B C Y	A T U W	C E G H I R S
(b)	9	A E K	A B L M T	B C D M S T V
(c)	10	A D I	A E R R	B E G M O R Y
(d)	11	A E L R	A D L M R	A B C I J K L
(e)	14	E N O	A G N O T	A C K M N T U
(f)	18	E H T	B E N O W W	A B N O P S T

Some Suggested Answers (frequently there are others):

```
(a)    C          (c)   AI R        (e)   NOT
       I                R               GONE
       T              DEAR                 N
     BAY

(b)     T         (d)    R          (f)      R
        A               READ                H E
     TABLE                A                WO N
        K              ALL                NEWT
```

If you enjoy this kind of puzzle, you may like playing ON-WORDS. The Game of Word Structures. Free information about this and other instructional games is available upon request from The Foundation for the Enhancement of Human Intelligence, 1900-W Packard Road, Ann Arbor, MI 48104.

THE VICAR TOLD THE SEXTON . . .

The sexton at the local church was ill and did not attend the Sunday service. The vicar visited him after the service, and the following conversation took place.

Vicar: "There were only 3 people in the congregation — excluding myself — and the product of their ages was 2450. The sum of their ages was twice your own age. Can you tell me the ages of the members of the congregation?"
Sexton: "You haven't given me enough information."
Vicar: "It is sufficient for you to know that I was the oldest person there!"

Now there is no doubt that the vicar was a very mathematically minded man; but how old was he?

. . .AND THE SEXTON TOLLED THE BELL

By the following Sunday the sexton had recovered well enough to be able to ring the bells at the regular service. There were just two bells in the church belfry; the first can best be described as a 'ding' — the second, not unnaturally, as a 'dong'. Now an ancient by-law in the district proclaimed that no 'ding' could be rung exactly two chimes after another 'ding'; and that no 'dong' could be rung exactly three chimes after another 'dong'. So what was the longest sequence of chimes that the poor sexton was permitted to ring?

Games & Puzzles

COMPUTER RECREATIONS

by D. Van Tassel

Calendars

Calendars provide some interesting programs. To program calendar programs you must know the following rule: A year is a leap year (that is, February has 29 days instead of 28 days) if it is a multiple of 4, except that a multiple of 100 is a leap year only if it is also a multiple of 400. January 1, 1800 was a Wednesday.

A common program on terminals is one that accepts any date and prints the day of the week. People like to input their birthdate and find out what day of the week they were born. This program can also be used to locate future three day weekends caused by holidays such as the 4th of July.

A common program needed in business programming is a program that accepts two dates and calculates how many days have passed.

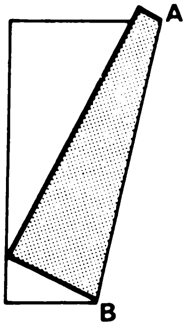
A more interesting problem is to figure out how many different calendars there are. The number is really quite small. It is possible to print a perpetual calendar. That is print all the possible calendars and then provide a table showing the years and the calendar to use for each year.

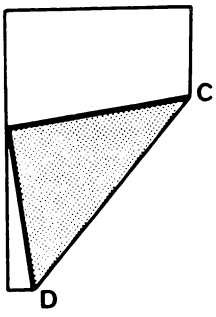
The final calendar problem concerns Friday the 13th. First calculate the probability of the 13th of the month being on a Friday. Next count how many Friday the 13th's occur in this century. Is the 13th of the month more or less likely to be a Friday than any other day of the week? Why?

FOLDED

A piece of paper may be folded so that the bottom right corner just touches the left edge. Your problem is to determine, for any rectangular piece of paper, a method to produce the shortest possible crease. In the diagrams, crease CD is longer than AB but still not the shortest. What is? If you rotate the piece of paper 90 degrees (longest side horizontal) and repeat the exercise, what is your method then?

Games & Puzzles





Thinkers' Corner

by Layman E. Allen © 1975

MATHEMATICS PUZZLES

How many of the problems (a) through (f) below can be solved by forming an expression equal to the GOAL? (Suppose that each symbol below is imprinted on a disc.)

The expression must use:
(1) only single digits combined with operators,
(2) all of the discs in the REQUIRED column,
(3) as many of the discs in PERMITTED as you wish, and
(4) exactly one of the discs in RESOURCES.

Special Rules:

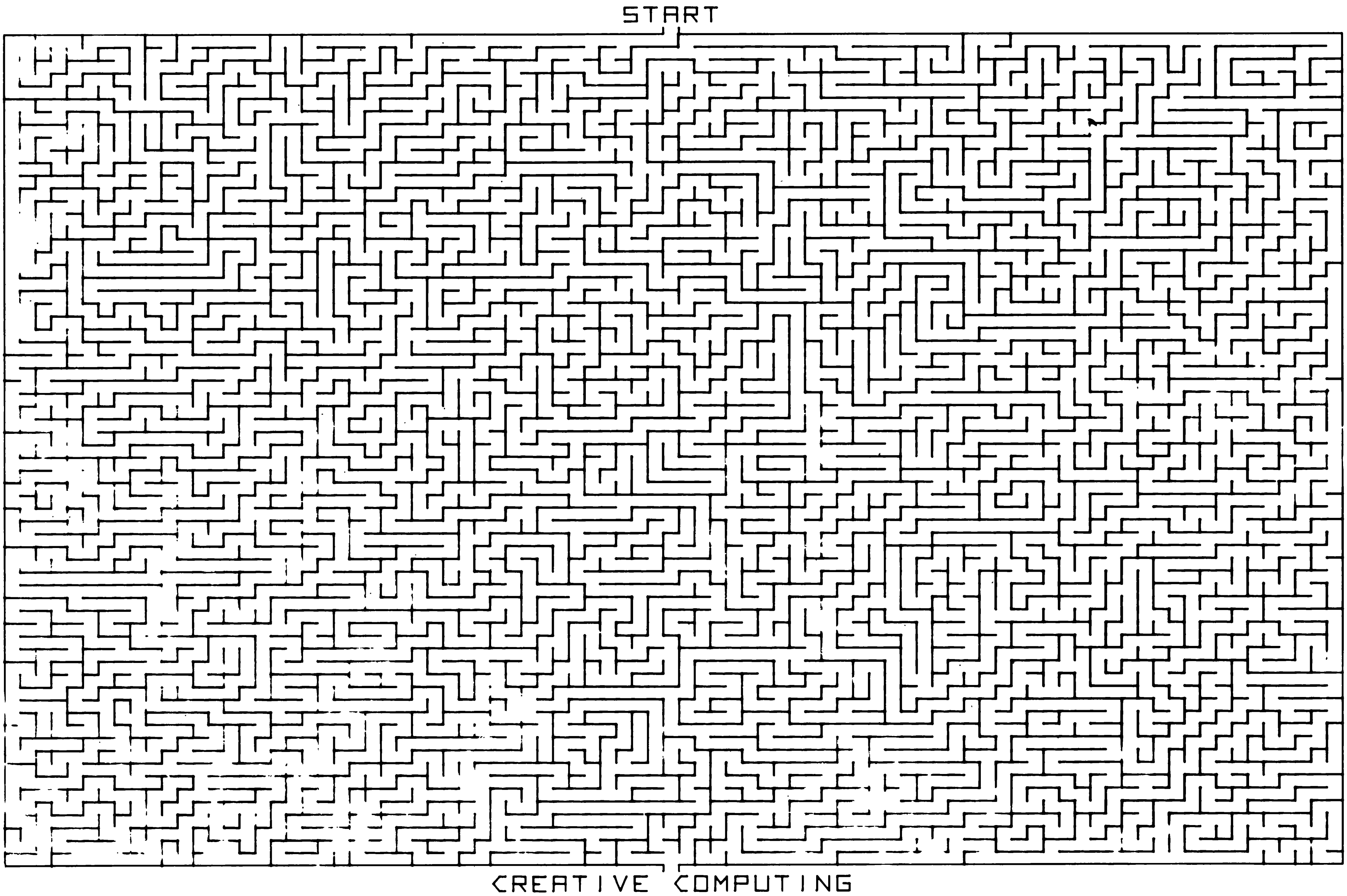
The '*' indicates "to the power of." Thus $3*2 = 3^2 = 9$.
The '√' indicates "the nth root of." Thus $^3\sqrt{8} = 2$.
Parentheses can be inserted anywhere to indicate grouping, but never to indicate multiplication.

Problem	GOAL	REQUIRED	PERMITTED	RESOURCES
(a)	11	67 -	13+	+ - ÷ √246
(b)	13	1 +	34-	X ÷ √ 1359
(c)	1	18 ÷	4+	+ - ÷ 1279
(d)	7	3-	248	- X ÷ √678
(e)	2	3 ÷	37- ÷	+ - X 0135
(f)	1	9√	22+	- X ÷ √- 28

Some Suggested Answers (frequently there are others):

(a) 7 + 6 - 2
(b) (4 X 3) + 1
(c) (8 ÷ 4) - 1
(d) 2 - (3 - 8)
(e) (7 ÷ 3) - (1 ÷ 3)
(f) 2 √ 9 - 2

If you enjoy this kind of puzzle, you might like playing EQUATIONS: The Game of Creative Mathematics. Free information about this and other instructional games is available upon request from The Foundation for the Enhancement of Human Intelligence, 1900-E Packard Road, Ann Arbor, MI 48104.



Here is a maze that was computed and plotted by a computer. The program that generated the maze was written in BASIC. It begins by asking for the name of the maze and then for the size (it may be up to 85 by 65). Each maze that is generated will be different if given a different name and/or size.

The program was written for the HP 9830A desk top computer and the HP 9866 plotter. The program generates the maze by making a random walk through the area until it can no longer walk. It then starts walking from another point. It walks until it has visited each and every room. The paths it took are plotted to form the maze. For a program listing, send your request with a large self-addressed, stamped envelope to: Gary D. Sasaki, Hewlett Packard, 5301 Stevenscreek Blvd., Santa Clara, CA 95050, (408) 246-4300

HOW LATE CAN YOU SLEEP IN THE MORNING?

David H. Ahl

Probabilities and expected values are a vital part to writing almost any game or simulation. Here are two real-life problem situations (I face them practically daily) which can be solved with simple statistics. Be warned: the second is considerably more difficult than the first.

GETTING TO WORK

Driving to work, you can take one of two routes. Route 1 is 5 miles long and has 4 traffic lights. Each light is on a 1-minute cycle but with different intervals. Light 1 is green in your direction for 30 sec., red for 30 sec. Light 2 is 20 sec. green, 40 sec. red. Light 3 is 25 sec. green, 35 sec. red. Light 4 is 40 sec. green, 20 sec. red.

Route 2 is 5.2 miles long with only 1 traffic light which is green your way 20 sec., red 40 sec.

Speed limit in the town is 35 mph.

1. Which is the best route and what is the expected time difference between the two?

2. Route 2 also takes you by a factory loading dock. If a truck is just arriving (occurs 1 day out of 30) you will be held up an average of 3 minutes. Does this change your answer?

AND ONCE THERE

Parking your car, you rush into the 10-story Morristown AT&T Building for your 8:30 am appointment with the vice president whose office is on the 9th floor. As you reach the elevators you glance at your watch and see that it is 8:29. From past experience you know that there is a 40% chance of an elevator stopping at any given floor; a stop takes an average of 10 seconds. The elevator passes from one floor to another in 6 seconds. There are 3 elevators all of which have an indicator on the 1st floor (where you are) that shows the location and direction of that elevator. At the point of your arrival, each elevator is equally likely to be on any of the 10 floors going on either direction.

1. Assuming you take the elevator, what is the probability that you will make your appointment on time? What is the probability that you will be less than 1 minute late? Less than 2 minutes late?

2. Under the conditions stated, exactly when would you expect to arrive on the 9th floor?

3. You also know from past experience that you can run up the stairs to the 2nd floor in 10 seconds. The 2nd to 3rd takes you 10% more time (11 sec.) and 3rd to 4th 10% more time (12.1 sec.) and so on. What position of elevators upon your arrival would cause you to run up all 9 floors? (Many answers are possible — select one "break even" combination).

4. Assuming your office is on the 3rd floor and you are faced with the same situation as above (time 8:29 with 8:30 appointment on the 9th floor), but with no elevator indicators, what is your best strategy to be on time for the meeting or as little late as possible? That is, do you run up or wait for the elevator?

PARTIAL ANSWERS

"Getting to Work." 1. Route 2 is approximately 0.47 sec. faster. 2. The arriving truck has an expected value of -6 sec. per day on Route 2, hence Route 1 now has the edge by 5.53 sec.

"And Once There." Once you get on an elevator, it is a fairly simple matter to determine how long it will take to get to the ninth floor (8 floors x 6 sec. = 48) plus (40% chance of stopping x 10 sec. per stop x 8 floors = 32) equals 80 sec. But figuring when the elevator will arrive on the first floor is something else again. Any elevator can be at any floor going in either direction. Hence, Elevator 1 has a 0.056 probability of being at, say, Floor 3 going up*. How long until it returns to Floor 1? Well, if we know it goes to the 10th floor, that's easy, but it has only a 0.4 chance of going to the 10th floor, 0.4 chance of going to 9 and so on. Multiplied by 18 different possible starting positions and by 3 elevators, this is a nasty problem. In a situation like this you have to ask yourself whether a heuristic, or rule of thumb, or best guess wouldn't provide an adaquate answer. For example, you might want to make the assumption that at least one elevator is at the 4th floor (or below) and heading down.

It is sometimes easier to come up with a solution if you think of the problem in entirely different terms. For example, think of the elevator as a one-way trolley on a circular track — station 1 is Floor 1, station 2 is Floor 2 going up, and so on. Station 18 is Floor 2 coming down and then back to station 1 again. Using this approach *may* make it easier to work the problem.

By the way, it should be apparent that a computer isn't much help in solving this particular problem. However, if this were part of a much larger simulation in which the output of one part provided the input to the next (a very typical situation), a computer would be almost vital to the solution.

If you're still with me and want to read a fun little book on the subject, get "Flaws and Fallacies in Statistical Thinking" by Stephen Campbell published by Prentice-Hall.

---

*If there are 10 floors and the elevator has an equal chance of being at any floor going in either direction, why isn't the probability of being at Floor 3 going up 0.1 x 0.5 = 0.05? Simply because Floor 1 and 10 do not have a direction associated with them, hence there are really only 18 locations. Actually, that's over-simplified because the elevator may not even reach Floor 10, or 9 or 8 etc. ■

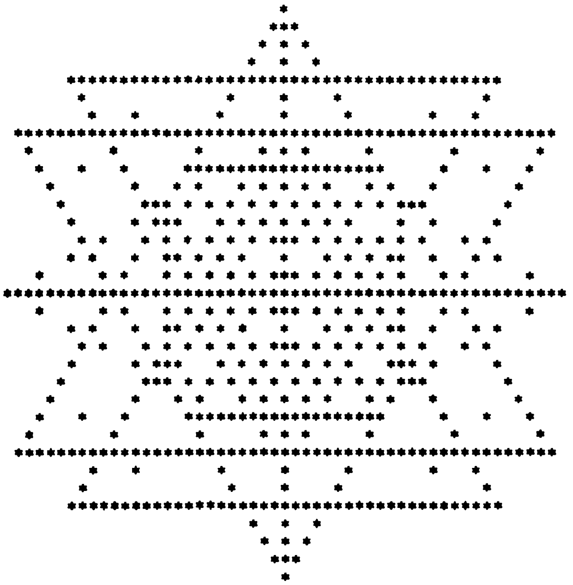
A Picture In 20 Lines

by E. Young
Beavercreek High School
Xenia, Ohio

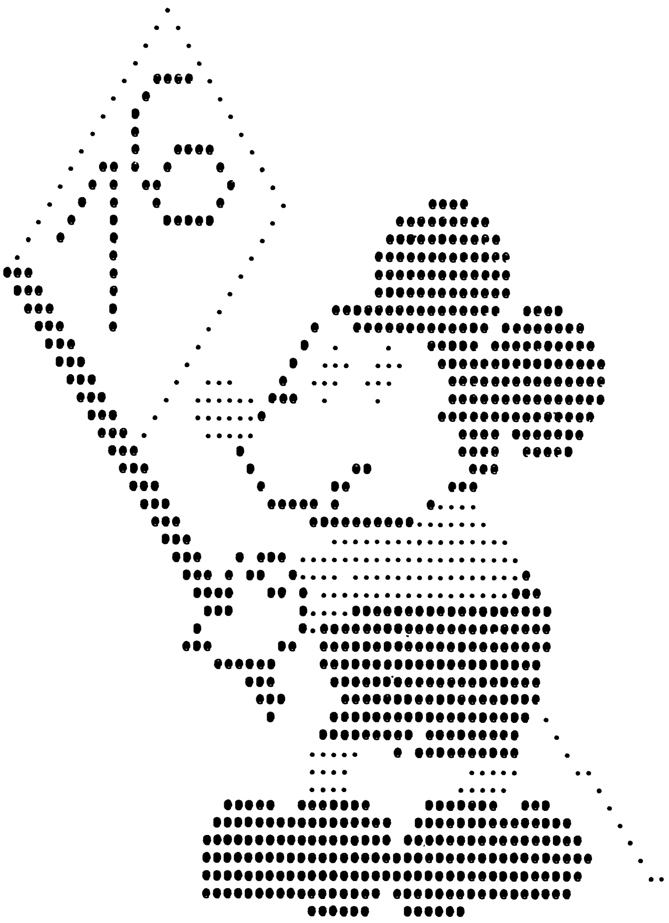
We've all heard that a picture is worth 1000 words. Well what kind of picture can be produced in a 20-line BASIC program (approx. 1000 characters).

My assignment to my first-semester computer science class was simply "to produce a picture with a 20-line BASIC program with no PRINT quote formats allowed."

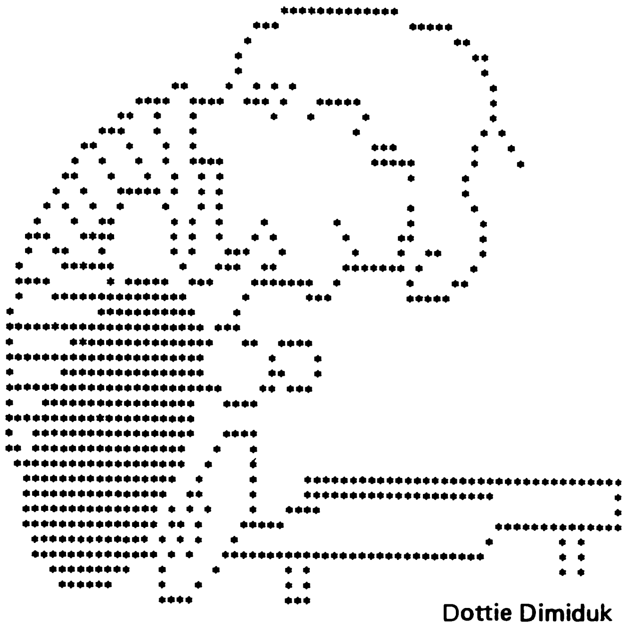
The variety of programming methods surprised me. They ranged from 3 data codes for what, where, and how many characters to read — to single numerical data that was sectioned algebraically to code a whole line.



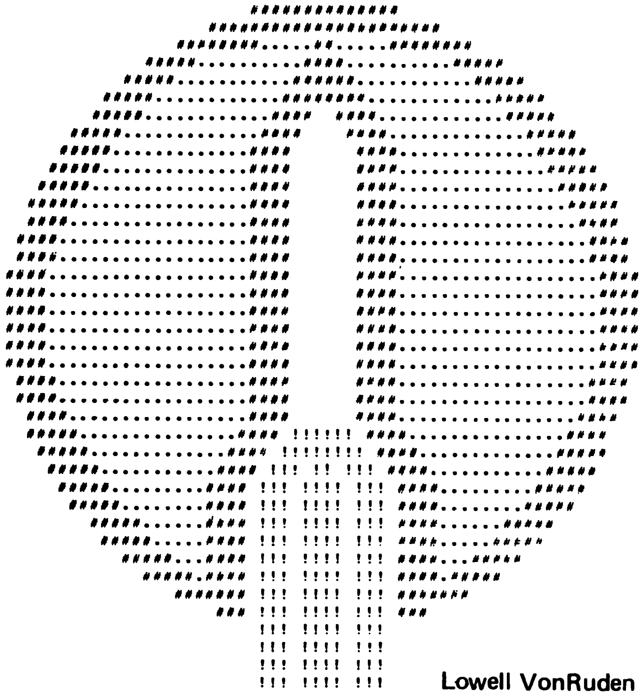
Star of Beaver Creek by Dave Triwush



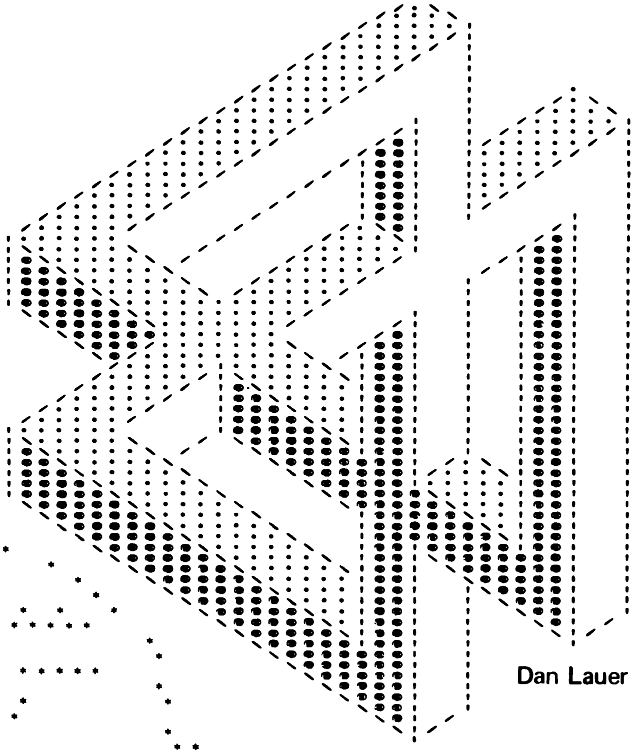
Susan Gordon



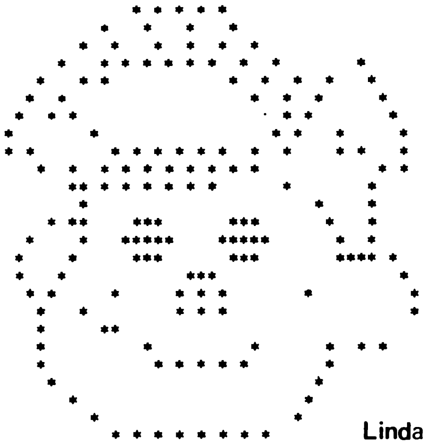
Dottie Dimiduk



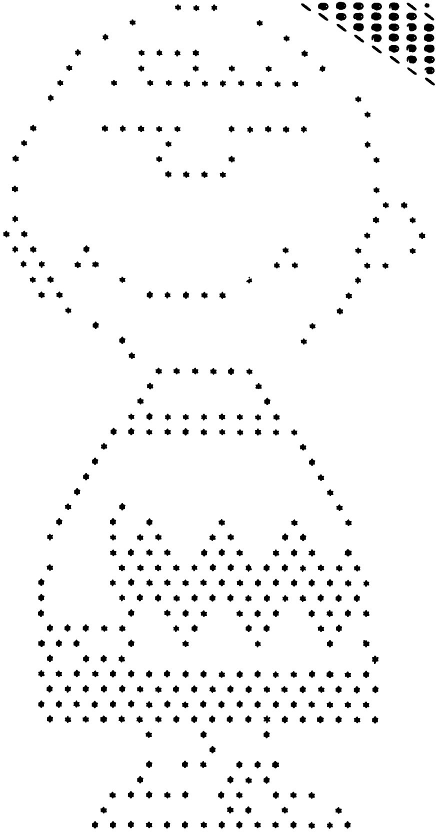
Lowell VonRuden



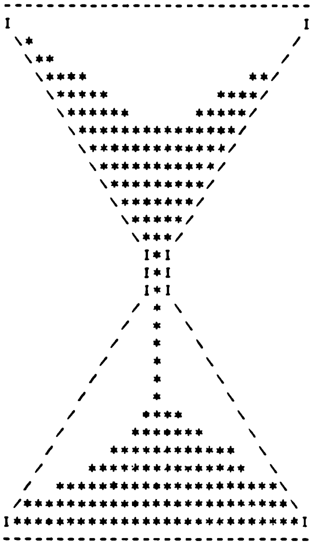
Dan Lauer



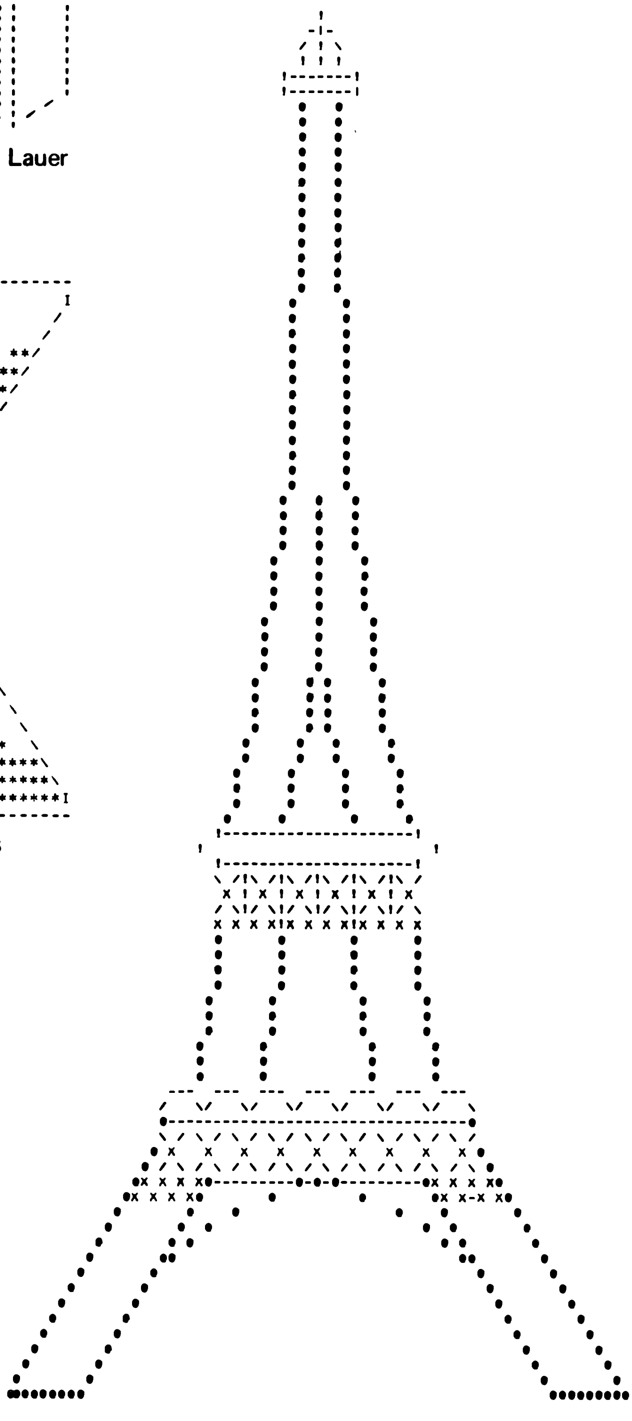
Linda Bailey

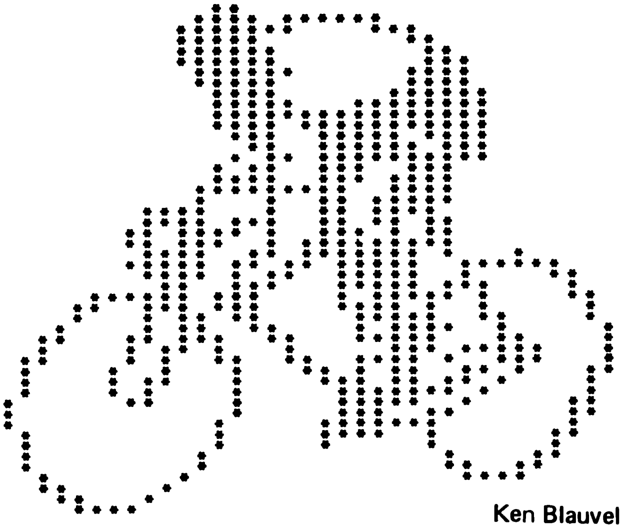


Pat Brunner

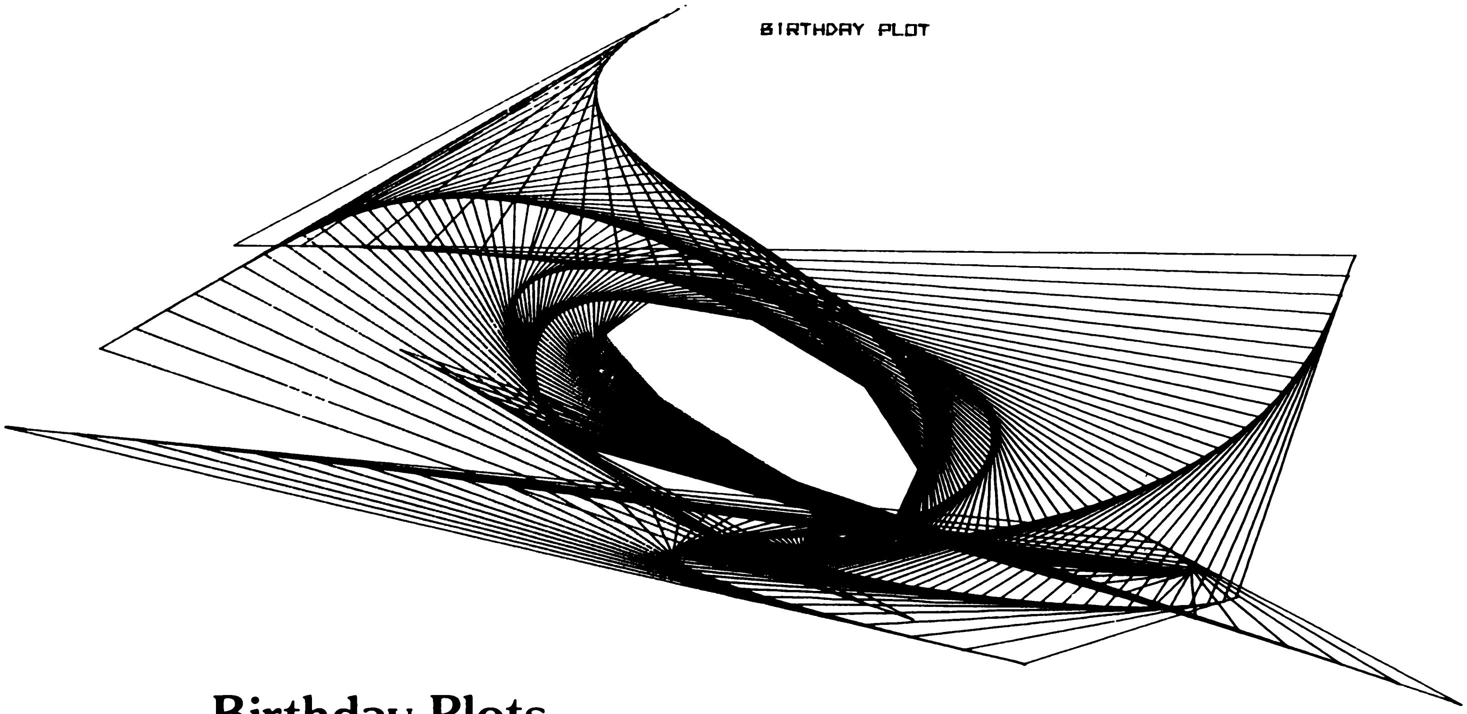


Norman Hicks





Ken Blauvelt

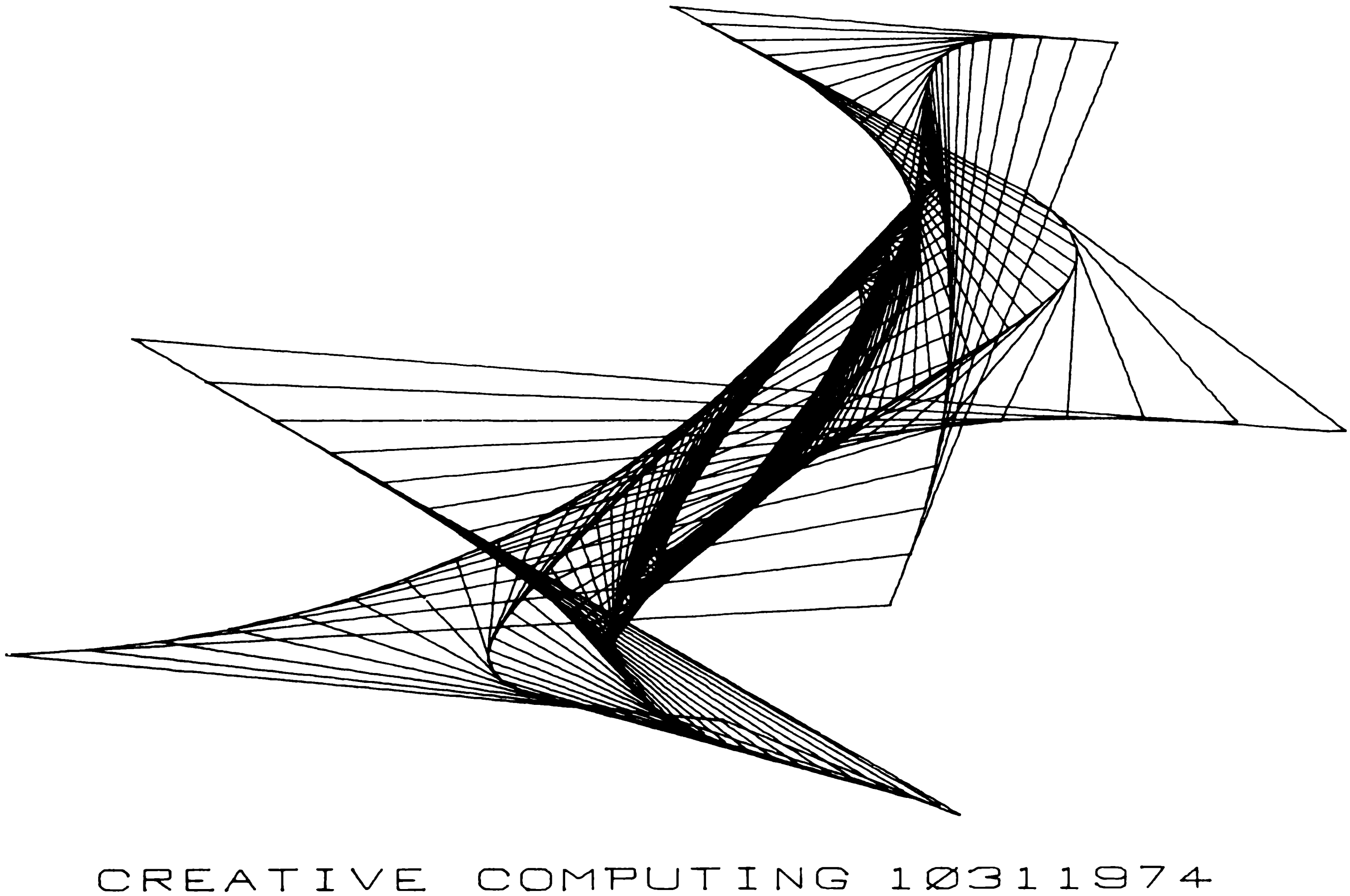

BIRTHDAY PLOT

Birthday Plots

At a recent math teachers conference, both HP and Wang were running "birthday plots." The two programs were remarkably similar in concept. Here are the two plots generated by the birthdate of *Creative Computing*, 10/31/74. All the lines in each construction are straight.

HP 9830 birthday plot

Wang 2200 birthday plot


CREATIVE COMPUTING 10311974

COMPLEX PROBLEM-SOLVING EXPERIENCE FOR UNDERGRADUATES VIA COMPUTER TECHNOLOGY

Michael Szabo, Ph.D.
Assistant Director
CAI Laboratory
The Pennsylvania State University
814/865-0471

and

Thomas Rhoades
Coordinator of Research
Anne Arundel County Schools

Scenario

Imagine the following scene: You sit down at a computer terminal and after brief introductory details, you are asked to read the following passage.

"The Arctic Ocean is bordered across most of Eurasia and North America by low, mostly flat plains that dip gently into the sea, called the tundra. With an average annual temperature of two degrees above zero the tundra remains frozen most of the year. When the spring thaws the snow in the higher lands to the south, the water runs into the river and begins a migration to the Arctic Ocean, but the mouths of these great rivers are still frozen solid in these cold, northerly reaches and present an impenetrable dam to the advancing waters. The water spreads out over the low landscape forming innumerable lakes, ponds and swamps. The water has no choice but to sit and wait for the thawing of the river mouth — it can't evaporate into the cold air, and it can't sink into the ground because it's permanently frozen to depths of hundreds of feet.

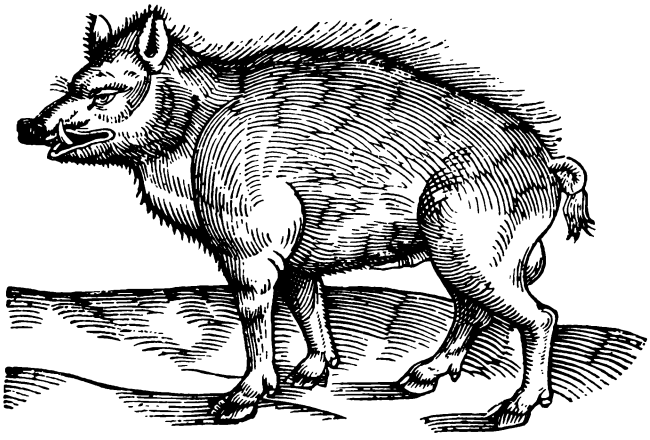
About one seventh of the entire land surface of the earth is permanently frozen and the greater part of that is covered with a layer, varying in thickness from a few feet to more than a thousand feet, of stuff we call muck. Only the top two or three feet of the muck ever thaws in the short arctic summer, only to freeze solid again during the winter. Ten feet down the passage of the seasons goes by completely unnoticed.

When the mouths of the great rivers finally open, much of the water drains away into the Ocean, carrying some of the muck with it. Then, occasionally, the excavated muck leaves to reveal one of nature's great oddities, the frozen remains of the great wooly mammoth.

The most famous of these, the Beresovka mammoth was discovered by a Siberian tribesman around the turn of the century. It was sticking headfirst out of a bank of the Beresovka River, a tributary of the large Kolyma which empties into the Arctic Ocean. A scientific expedition was sent by the National Academy of Sciences from St. Petersburg in accordance with an edict from the Czar. The members of this company built a shack over the corpse and lighted fires within to thaw it out. They then dismembered it carefully, packed up the parts, refroze them in the air outside, and sledded them to the Trans-Siberian Railroad.

Much of the head, which was sticking out of the bank, had been eaten down by wolves and other local animals, but most of the rest was perfect. Most important however, was that the lips, the lining of the mouth and the tongue were preserved. Upon the tongue as well as between the teeth, were portions of the animal's last meal, which for some reason he had not had time to swallow.

This meal proved to have been composed of delicate sedges and grasses and — most remarkably — of fresh buttercups in flower. The stomach contained many more quarts of the same material, undigested and not decomposed. The exterior of the mammoth was whole and perfect with none of its two foot hair rubbed or torn off. The meat was fresh, although it started to rot when thawed, and the sled dogs ate it with enthusiasm relieving the party from carrying additional supplies for the dogs. Unlike most mammoth finds the Beresovka mammoth had a broken hip, but like others it was healthy and robust, upright and with no apparent cause of death.



Mammoth remains are found widely distributed over Siberia and in Alaska. Their bones are spread over the floor of the Arctic Ocean where ships have dragged them up. And, 200 miles to the north, in the New Siberian Islands, mammoth remains are thickest of all.

How does one go about killing healthy, robust — 5 ton — animals leaving no apparent cause of death and then manage to preserve them in the condition described using only natural means? What processes of nature could combine to concoct such a wild story?"

The above scenario introduces you to a unique and rare experience in an undergraduate curriculum: the opportunity to attempt to solve a real and complex problem in science. The experience, which is supplemented by a computer, is also a vehicle to study human problem-solving of a scientific nature.

Scientific Problem-Solving

Although scientists and psychologists have long inquired about the nature of how man solves problems, most of the studies have focused upon the solution to problems with single correct answers. It can be argued however, that the study of how problems are solved can yield much insight into how man thinks. Given the assumption that problem-solvers employ a wide range of skills and strategies, complex problems are needed to elicit those behaviors in order to detect and study them.

We might define a complex problem as one having two characteristics. First, a rational (i.e., scientific) approach may be employed in the search for a solution. Second, the problem is rich in data which implies that various possible conclusions can be supported, to some degree, by these data.

The shape and direction of our computer-supplemented study of the Mammoth continues to be guided by these basic questions.

> How does one solve complex problems?
> Can a person become a better problem-solver?
> What discriminates the better problem-solver?
> What environments facilitate problem-solving?
> To what extent can the study of problem-solving be automated?

Overview

The Wooly Mammoth (Mammo) problem is a problem-solving environment designed to confront individual or small groups of problem solvers with a complex task. The role of the computer (an IBM 1500 series using Coursewriter II with CRT, image, and audio terminals) is to: 1) present the problem to the student; 2) serve as a data source; 3) monitor selected parameters useful in the study of problem-solving; and 4) provide statements (heuristics) which help structure the flow of the solution effort.

Dr. David Riban conceived Mammo during his tenure as a physics graduate student at Purdue. Although designed for use with a high-speed computer, Mammo was run numerous times in non-computer mode with high school physics students and undergraduates. Mammo was programmed for the 1500 System in 1973 by Mr. Thomas Rhoades and Dr. Michael Szabo. Since that time it has been used to provide over 200 student hours of training in complex scientific problem-solving to preservice secondary science teachers. In addition, more than 350 undergraduate student hours have been logged in conjunction with research studies on problem-solving conducted by Dr. Szabo, Mr. Rhoades, Mr. Lazzaro, and Melissa Berkowitz.

Operation

An operational flowchart of student progress through the two-hour Mammo sequence is presented in Figure 1. The program first handles housekeeping details and presents the Scenario to the student. Then, without identifying the problem to be solved, the computer requests possible hypotheses and suggests the student query the computer (request data) that might be useful in testing the hypotheses. At no time does the student see a listing of available data. Instead, he types in a natural language request (up to three lines of text) which is matched with stored data titles using an edit and a keyword routine.

For example, a student might request, "What was the age of the mammoths?" This simple statement is deceptive. Does the student wish to know the number of years since mammoths became extinct? the actual age of the Beresovka Mammoth at death? the actual age (average) of mammoths at time of death in general? This example points out: 1) difficulties of using a computer to assign semantic meaning to a string of characters and; 2) the tendency of problem-solvers to formulate imprecise data requests. A successful data match results in the display of verbal data on the image screen. An unsuccessful match triggers the data request sequence again, but in a slightly different form. The student is given the three options of making a final conclusion and ending the program, reviewing hypotheses, or making another data request.

The data request used as an example will trigger display of a brief passage regarding the geologic age of the mammoths, abstracted from the scientific journal *Science*. There are more than 200 separate pieces of data stored on accessed images.

This describes one version (Mode I) of Mammo, called the Problem-Mode. Mode I was designed to be free from hints or cues, both content-oriented and content-free, that might aid the student in solving the problem. A second mode (II) has been constructed to provide content-free problem solving heuristics to guide, shape, or direct the process used by the student.

Operationally, Mode II is identical to Mode I except for two added features. The first is quite simple: it forces the student to evaluate his progress (hopefully, to test hypotheses with the data acquired) by requiring that the student type a brief conclusion after he has received data.

The second feature unique to Mode II is the presentation of selected heuristics to "guide" the problem-solving approach used by the student. By responding to the pattern of data requests or conclusions posed by the student, Mode II heuristics simulate a real problem-solving environment. Two examples will be presented. If the student triggers three pieces of data wihch have been determined to be logically related (by an outside authority), he receives the following heuristic message:

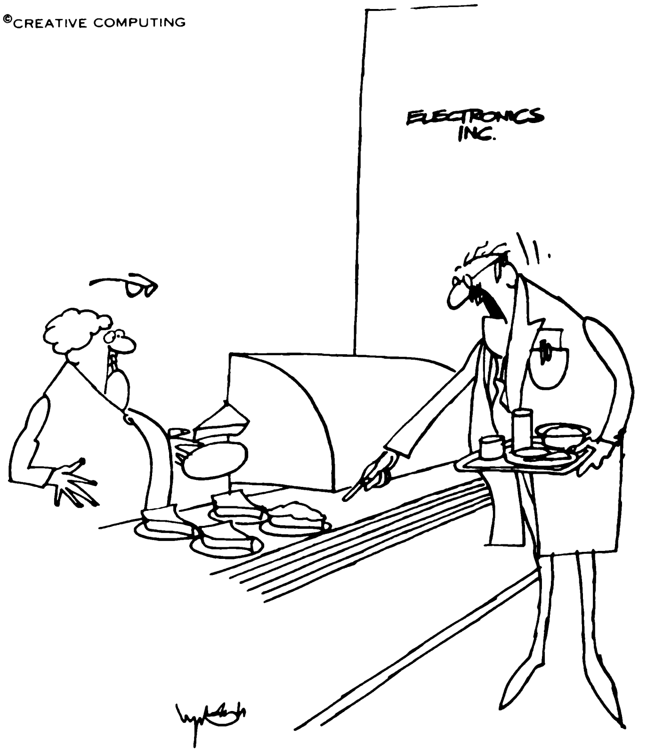
> "Good. Your last few requests for data show that you're following up a single line of thought. This is the only profitable way to attack a complex problem like this."

If on the other hand, the data requests (and matches) indicate logically unrelated data being sought, the heuristic may be of the following form:

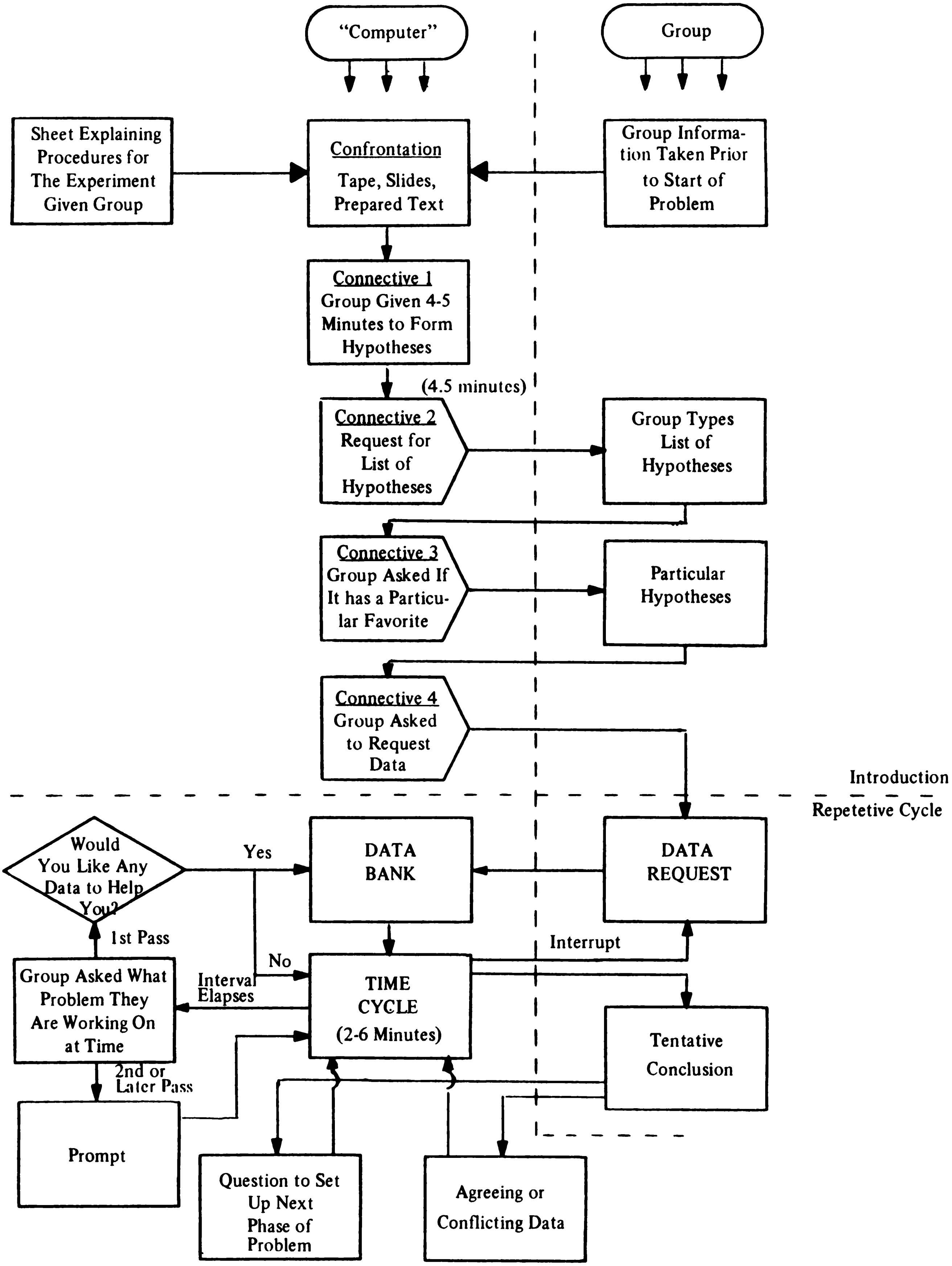
> *"Do you have some strategy or plan of attack on the problem? What are you really trying to clarify?"*

As another example of the second unique feature of Mode II, the conclusions made by students are compared

©CREATIVE COMPUTING


ELECTRONICS INC.

". . . Pi!! . . . . That's the quantity I forgot to factor in!!!"


"Computer"
Group
Sheet Explaining Procedures for The Experiment Given Group
Confrontation
Tape, Slides, Prepared Text
Group Information Taken Prior to Start of Problem
Connective 1
Group Given 4-5 Minutes to Form Hypotheses
(4.5 minutes)
Connective 2
Request for List of Hypotheses
Group Types List of Hypotheses
Connective 3
Group Asked If It has a Particular Favorite
Particular Hypotheses
Connective 4
Group Asked to Request Data
Introduction
Repetetive Cycle
Would You Like Any Data to Help You?
Yes
DATA BANK
DATA REQUEST
1st Pass
No
Interrupt
Group Asked What Problem They Are Working On at Time
Interval Elapses
TIME CYCLE (2-6 Minutes)
Tentative Conclusion
2nd or Later Pass
Prompt
Question to Set Up Next Phase of Problem
Agreeing or Conflicting Data

SIMPLIFIED FLOW DIAGRAM FOR MAMMO

with prestored conclusions which have been posed by scientists who studied the riddle of the mammoths. Further, logical "chains" or sets of data which are necessary and sufficient conditions to support these conclusions have been identified.

Presume that a student concludes that the mammoths froze to death but the student has neither requested nor seen the necessary data "chains" to support this conclusion. The likely computer heuristic displayed to the student, depending on specific terminology used is:

> *"A comparision of all possible facts and data in memory areas implies you have seen insufficient data to form this conclusion.*

On the other hand, suppose the student has requested the necessary data and concludes the mammoth frozen to death. In this case, the heuristic is:

> *"Hold on! Let's go back and check out your conclusion about the mammoth's death. Ask yourself questions like — if the mammoth really died in this way, what conditions would the cadavers be in? Always check your conclusions. Consider the idea and determine what data you need to check your idea out, then request such data."*

From these heuristics, it can be seen that the intent of the guidance provided in Mode II is to urge the student to form hypotheses, devise critical tests of these hypotheses, request data needed for those critical tests, and apply the tests in some logical fashion to reducing the discrepancy between the problem statement and the solution statement.

Findings

In addition to specific and experimental findings from theses now in progress, the following observations represent a sample of those made by various individuals associated with the project.

First, attempting to solve Mammo is psychologically engaging and elicits some strong positive reactions. Most students have to be reminded that it is time to quit and a large majority wish to know "the answer." The enjoyment of the experience is a function of the background (science or non-science) of the students and whether the problem is investigated in a group or as an individual effort.

Second, requests for data are more often requests for conclusions. For example, students commonly ask questions like — "Did the mammoth drown?" rather than requesting data on analysis of the stomach contents of the cadaver (that is, data which could support or refute the hypothesis of death by drowning). The phenomenon of imprecise or unfounded data requests has been noted in many problem-solving studies and was a serious deterrent to the acquisition of sizable amounts of data in early Mammo programs. This problem has been corrected by the addition of a short segment which instructs the student to conceptualize the computer more like an encyclopedia than a genie with all the answers.

A third observation is that terminology used in requests for data varies considerably, with profound implications for keyword identification of the data. Use of common synonyms for the Lorge-Thorndike dictionary of most commonly used words was quickly found to be unworkable as college students just didn't use those terms. An examination of actual terminology used led to substantial revisions which in turn permits much greater access to data.

A related problem is serendipity. Occasionally a student request triggers the display of data apparently not intended. Early in the development, a data request which included the words "weather" and "mammoth" would trigger data on the eating habits of mammoths. Although the student apparently desired climatological data, he actually obtained data on eating habits of the mammoth. The problem was soon discovered: the word — eat — is embedded in the word weather. Although this problem of incorrect matching of requests with displayed data results from limitations on CAI language restrictions and programming skills, it nonetheless simulates reality to some degree. The researcher always encounters data which: 1) he wasn't looking for and 2) are irrelevant to the hypotheses presently under investigation. The effect of unwanted but potentially useful information upon a student's problem-solving strategy might be of considerable value.

Utilization of Mammo

Past and future utilization plans for Mammo include the: 1) investigation of problem-solving skills and strategies (e.g., problem conceptualization and reconceptualization, hypothesis formation, data utilization); 2) study of problem-solvers' behaviors in various prompting (environmental modes) modes; 3) comparsion of group and individual problem-solving behaviors; and 4) training of prospective and inservice science teachers in problem-solving skills.

Presently two doctoral and one masters thesis are in progress as are minor revisions to the program.

A second complex problem is being developed in order to provide an alternative format and serve as a criterion environment to investigate increases in problem-solving abilities or performance levels.

Summary

This paper describes a system of complex and scientific problem-solving in its most recent (but hopefully not final) stage of evolutionary development. A real world problem has been partially programmed into a computer program with two major modes. One presents the problem and assumes the student proceeds toward a solution in a more-or-less logical (i.e., non-random) fashion. The second mode attempts to guide the student into using certain strategies theoretically related to scientific problem-solving.

The complex answer processing routine which borders on an artificial intelligence application simulates numerous aspects of real problem solving including the fact that the student never knows the complete domain of available data.

Mammo has been used to train prospective science teachers and to study selected parameters of human problem-solving. It also serves as a vehicle to test the limits of technological devices in recognizing specific human thought patterns and responding intelligently to them. ■

"Quick, Charlie! How much is 27 minus 6?"

©CREATIVE COMPUTING

Learning by Doing

by Fred Gruenberger

In a second, or intermediate, course in computing, the student should be lured (coerced, dragooned) into working on a term project, assigned at the start of the semester and due at final exam time. This should be a computer problem, preferably of the students' own choice, done in workmanlike manner to demonstrate that the student has learned something of the computing art. The final result should be packaged neatly, to include:

1. An English statement of the problem.
2. Flowcharts of the logic of the solution (or the equivalent of flowcharts, if the student prefers some other way of expressing his logic).
3. Listings of the program.
4. Results, clearly labelled.
5. A test procedure and test results.
6. Conclusions (what was learned from the work; what would be done differently if the project were to be repeated; limitations on the results; suggestions for further research, etc.).

   The packaged term project should be saved for use in eventual job interviews.

Experience has shown that the student's chief problem is that of selecting the problem he intends to work on. In all likelihood, he has so far worked only on problems that were assigned (and hence clearly labelled computer problems), for which much of the analysis has already been done for him.

There is usually a fair amount of panic at the time this assignment is made, since it puts the student on his own for the first time. About a third of a typical class will suggest one of the following:

1. "May I use a problem I did in my Fortran class last semester?" No, that problem was defined and analyzed for you, and the object now is to see how you do with a new problem. Further, you *did* that problem; it's time for you to do another one.

2. "I've been assigned to a big problem at work; may I turn it in for this project?" No; that problem concerns your work; try an isolated problem here, one that you can deal with thoroughly and completely. (And experience has shown that when this restriction is relaxed, the end result always seems to be someone else's work, and the student can barely explain what went on.)
3. "I don't know where to find a problem to work on." Well, there are systematic collections of good problems; you might try browsing through *Problems for Computer Solution* (Gruenberger and Jaffray, Wiley, 1965) which outlines some 90 problems that would be suitable. The best problem is the one that interests you.
4. "I'm a business (music, chemistry, ..., mathematics) major; I'll do a problem in business (music, chemistry, ..., mathematics)."

   There's the real problem. What is needed, early on in the semester, is a *proposal* by the student of just what he intends to do, so that he can be saved from the extremes of plunging into a problem that is either trivial or too grandiose. In the former case, he will produce something that requires little or no knowledge of computing; in the latter case, he will look sad at final exam time (when the computing center is saturated) and wail that he needs just one more run of his program.

   The phrase "a business problem" is rather vague. A new attack on Bill of Materials scheduling? An inventory control program for 10,000 line items? A table of base pay times hours worked? Or what?

The trouble is, of course, that the whole idea is brand new to most students; he has never been placed in the awkward position of making a selection that is within his own capabilities (indeed, he has probably never been told what constitutes a decent computer problem). Further, he has never had to define a problem and outline an attack on it; this has been done for him for all of the 14 years he has been in school. It helps if he can see samples of what this is all about, so a collection of old term projects (both good ones and bad ones) should be made available to him. It would help even more if he could be shown some sample proposals. A few are given here.

I

A comparison will be made between the Gauss-Seidel and Gauss elimination algorithms for the solution of simultaneous linear equations. A number of sets of six such equations will be constructed with known roots. Programs will be written in Fortran to solve such systems with the two algorithms, both in single and double precision. The solutions will be compared for the following:

1. Computation time.
2. Accuracy of the results.
3. Compilation time.

One of the sets of equations will involve a singular matrix, to determine how this condition is handled by the programs.

II

A small inventory of 25 items will be set up, and daily changes to that inventory will serve as input to a program. The program is to update the inventory, and print a report showing, for each line item, the quantity on hand, items out of stock, reorder conditions, lead time to arrival of new stock, and those items requiring expediting. For the small inventory involved, all results will be hand calculated as a test of the logic.

Reprinted with permission from *Popular Computing*, Vol. 3, No. 11. Copyright 1975 *Popular Computing*, P.O. Box 272, Calabasas, CA 91302.

III

The COBOL reference manual at our installation lists 58 error messages for errors that can occur at compile time. A program will be written that will trigger each of these error messages.

IV

The melody of a song can be expressed as a series of numbers. The pitch of each note can be expressed by numbering the notes of the scale, and the duration of the note can be coded on a scale from 1 to 8. With a given melody so coded, an algorithm can be applied to it to translate it into a new melody. The simplest such algorithm would be to reflect each note around a middle value, to transform high notes into low notes and vice versa, leaving the duration of each note fixed.

Several algorithms will be devised, and applied to ten "standard" tunes. The resulting translated melodies will be played, and the most pleasing result will be recorded and submitted on a cassette as part of this project.

V

The Los Angeles *Times* prints about 500 column-inches of text each day in its main news section. Some of these inches can be identified as politically oriented:

A. Favorable to Republicans.
B. Unfavorable to Republicans.
C. Favorable to Democrats.
D. Unfavorable to Democrats.
E. Favorable to third-party candidates.
F. Unfavorable to third-party candidates.
G. Completely neutral.

In theory, material that reflects the political leanings of the newspaper's editors should be confined to the editorial pages, and the news pages should be unbiassed. In practice, the amount of text space allotted to a candidate or a party reflects the paper's views, however unconsciously.

The pages of each issue for the 8 weeks preceding the last general election will be examined, and a listing made of the column-inches in each of the first four categories given above, as objectively as possible. Ratios will be calculated of the following:

$$\frac{A}{B} \qquad \frac{C}{D} \qquad \frac{A}{C} \qquad \frac{A+D}{B+C}$$

for each day, and progressive totalled for the 56 days.

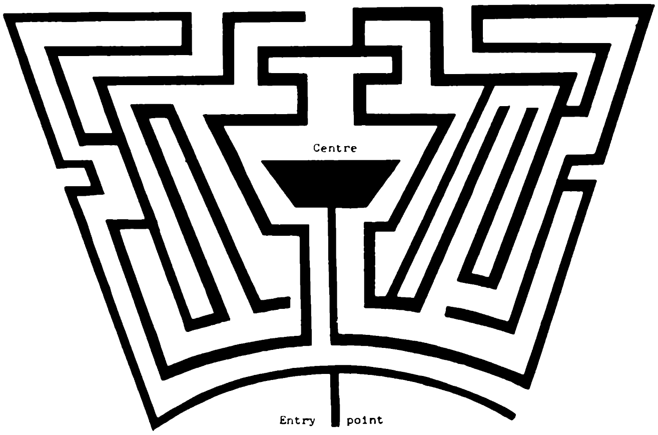
While this is not properly a computer problem (the small amount of arithmetic involved could easily be done by hand), the program will be useful for a much larger project jointly sponsored by the School of Journalism and the Department of Political Science. During the 12 weeks preceding the coming presidential election, the ratios will be calculated and plotted for each of 15 leading newspapers across the country.

VI

Attached is a diagram of the maze in the gardens at Hampton Court Palace, constructed in the reign of William III. "The key to the centre is to go left on entering, then, on the first two occasions when there is an option, go right, but thereafter go left." The maze exists today, and for 2p a visitor may enter the maze, seek the centre, and retrace his way out.

Given a new maze, described in terms of the coordinates of the branch points, a program is to be written to explore the maze and output the directions for the choices to be made to proceed from the entry point either to the center or to a specified exit. The choices will be limited to two at each branch point.

One test of the program will be the reproduction of the directions quoted above for the Hampton Court maze. The computer solution for the other mazes will be checked by hand.



Plan of the maze at Hampton Court Palace, England. The path to be followed is the black line.

VII

If the numbers from 1 to 20 are permuted, what is the distribution of runs up and runs down of the numbers? Consider the following possible permutations:

A	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
B	1	20	2	19	3	18	4	17	5	16	6	15	7	14	8	13	9	12	10	11
C	20	18	16	14	1	3	5	7	12	10	8	6	4	2	9	11	19	17	15	13
D	2	12	17	4	13	10	1	5	15	19	20	7	16	14	6	18	3	8	9	11
E	11	19	2	7	12	1	8	6	18	13	15	20	17	10	16	14	3	4	5	9
F	1	15	14	2	16	13	3	17	12	4	18	11	5	19	10	6	20	9	7	8

For A, there is just one run up, and none down, but for permutations taken at random, the chance of arrangement A is just 1 in 20!, or about 1 in 2.4 times 10 to the 18th power.

For B, there are 10 runs up and 9 down, but this, too, is an unlikely arrangement.

For C, there are two runs up and three down.

For D, there are six runs up and five down.

For E, there are seven runs up and six down.

For F, there are seven runs up and six down.

The complete distribution of all possibilities for all 20! permutations could be determined by theory. I propose to approximate the distribution by sampling random permutations and counting the runs.

In order to do this, I will need a random number generator and an algorithm for forming random permutations. For the former, I will use the generator described on page 8 of issue 21 of *Popular Computing* (my work will be done in Fortran). For the latter, one of the following schemes will be used:

1. Generate an array of 40 numbers. In the even positions put the numbers from 1 to 20. In the odd positions, generate 20 random numbers. Then sort the 20 pairs, using the random numbers as the sort key. Although it is inefficient, I will bubble sort the 20 pairs. After sorting, the right hand number in each pair of numbers will be a random permutation of the numbers from 1 to 20.
2. Use the random number generator to generate integers in the range from 1 to 20, and let these integers be the subscripts for entries in an array of dimension 20. First zero out this array. Select elements in the array by the choice of subscript. If the chosen element is zero, fill it with the next consecutive integer, starting with 1. If the element is not zero (i.e., it has already been selected), proceed to another element. When all the elements are filled, the array contains the desired random permutation.
3. In the above scheme, the first 16 or so elements will be filled rapidly (that is, the condition of duplicates will not occur too often), but the last 4 or so may take an undue amount of time. A variation might be tried. Apply the scheme described until 16 elements are filled. Then insert the remaining four numbers into the four blank slots, picking one of 24 arrangements of those four positions at random, again using the random number generator.

At least 1000 random permutations will be generated, and a distribution made of the lengths of the runs. This distribution will be compared to the theoretical (if I can obtain that). The program will be generalized so that it can operate on dimensions other than 20. If time permits, runs will be made on permutations of 10 and 30 items.

VIII

I would like to try to calculate the number (15000!). (I understand that the current record for factorials is 10000!, and that all the factorials by thousands are on file.) Even if I don't succeed in obtaining the desired result, I believe that working on the project will be worthwhile, and I will be able to at least provide hints and suggestions for the next person who tries to break the record.

Since this calculation will consume considerable amounts of machine time, I propose to calculate the following test data first:

1. The exact number of expected digits.
2. The number of low order zeros.
3. Some of the high order digits, and some of the low order non-zero digits.

In addition, I will use my program to calculate (and check with known results) 500!, 1000!, and 10000! before requesting a commitment of machine time for the long run.

I believe that I can hold intermediate results in storage by packing six decimal digits per machine word. I will need packing and unpacking subroutines, and a subroutine for decimal multiplication. I will test these subroutines before making any long runs.

IX

Problem H5 in *Problems for Computer Solution* calls for the creation of abstract paintings by a computer program. The program is to select the size and shape of various geometric figures and their position on a canvas. I wish to explore this notion extensively with the aid of the plotter now available, which should aid greatly in the mechanical chore of examining the program's results. The plotter will allow up to three primary colors for the figures, and the choice of color will also be made by the program. The plotter routines also permit the figures to appear in outline form, or solid (filled in), as well as various degrees of cross-hatching.

It is stated in Problem H5 that an important aspect of the problem is the determination of when to stop. I propose to put this decision into the program in terms of the area covered by the random figures. This will be a bit tricky, since the figures overlap. However, if the total area covered by the figures is limited to some fraction of the available area, with or without overlap, and this limit is a parameter of the program, then the program will be able to output finished art without intervention in any quantity.

The ratio of successive terms of the Fibonacci sequence approximates the golden mean:

$$\frac{1 + \sqrt{5}}{2} = 1.6180339887498948482045\ldots$$

Thus, we see

$$144/89 = \underline{1.61}79775$$

$$1597/987 = \underline{1.61803}44$$

$$121393/75025 = \underline{1.618033988}6$$

Using the EXACMATH package, I propose to write a program to generate successive terms of the Fibonacci sequence, form the ratio, and determine to how many digits the ratio agrees with the golden mean. The limits of the EXACMATH package will let me carry these calculations to at least the 1000th term of the sequence. My output will be a table of values (term number against the number of digits of agreement) and a graph showing the rate of growth of the function being explored.

X

The Raindrop Problem, which appeared in issue 6 of *Popular Computing*, called for selecting a point at random in the unit square as the center of a circle whose radius is taken at random between zero and 1/2 unit. The problem asked for the number of such circles needed to completely cover the unit square.

A crude solution will be attempted by subdividing the square into 400 smaller squares. A mathematical test can be devised to determine whether or not each of these smaller squares is covered by one of the circles. The results of 100 trials will be plotted, to obtain an approximation to the desired distribution. For a few of these trials, the method will be repeated with the large square subdivided into 900 smaller squares, to determine if the method could lead to a correct solution. ■

Could this be the answer to the need for a generalized form to present programs?

ALGORITHMIC BASIC

by Tom Allen
P.O. Box 81
Stevensville, MI 49127

Ed. Note: Recently I received a long letter from Tom Allen, most of which is reproduced below. If readers have reactions to the ideas put forth by Tom, please write him directly or send your thoughts to me at Creative Computing — DHA.

I read the letters to the editor and the "On Computer Languages" article in the Sept.-Oct. issue with a great deal of interest. It occurs to me that the question of the "proper" language for the *Creative Computing* context may be an example of confusing the medium and the message. After all, a particular program, written in a particular programming language, is only an implementation of a much more general solution algorithm. No well-designed algorithm depends on a specific language of implementation for its success. Moreover, it seems to me that a clearly-stated algorithm would be useful to a much wider group of potential users than would an implementation in some specific language. For instance, the primary language of the realm in which I currently reside is COBOL, in an RJE environment. So, in order to gain access to any of the programs you list in *Creative,* I have to extract the algorithm from the BASIC program, clean it up, and then implement it in COBOL. That's a'lotta work!

By way of translating these thoughts into something constructive, I have a suggestion to offer. Perhaps *Creative* could include a regular monthly column featuring the algorithm underlying some program appearing elsewhere in the issue or a program of wide-spread interest. Since I've spent a little time thinking about this idea, let me share some of those thoughts with you. First, it's a half-baked idea; the part that's still raw is the question of what language or meta-language should (or could) be used to state the algorithm. Tho there is an historic precedent for the use of ALGOL, my personal opinion is that this is a case of the cure being worse than the disease. And I don't think the graphic flowcharting language is the answer because of problems in trying to shoehorn a flowchart into an 8½x11 format for printing. It seems to me, however, that the so-called pseudo-language which the structured programming boys have been fooling around with lately might just be the answer. Using this language invites the additional use of the top-down, structured approach to the design of the algorithm as an additional plus. Anyway, I think that the problem of a language for the statement of algorithms can be dealt with reasonably well.

Getting algorithms out in the open so everybody can see them can produce another benefit as well. When the entire readership of *Creative* can see the algorithm, it then becomes possible for everybody to contribute to the refinement and generalization of an algorithm. As an example of this, I've been fooling around with a COBOL version of the Lunar Lander. Once I got the basic algorithm for the Lander out in the open, I began to see all sorts of extensions and generalizations that could be gotten almost for free with a suitable statement of a top-down, structured version of the algorithm. It also allowed me to "segregate" the physics behind the simulation in a very comfortable and understandable way.

The one facet of this idea which bothers me somewhat is a concern about the problems which the "typical" *Creative* reader/programmer might encounter in converting an algorithm into the program. The syntactical structures available in such languages as COBOL, ALGOL, and PL-I lend themselves easily to the implementation of a top-down, structured algorithm. This is less true of FORTRAN, and even less so of BASIC. Moreover, if one is using BASIC in a minicomputer environment (where else?), it is often necessary to be "tricky," and parsimonious in one's coding techniques so as not to exceed core limitations. Sometimes it's damn difficult to be both elegant and stingy!

So there's the idea, for what it's worth. I'd be most interested in your reactions.

One last item: do you know of anybody whose got an algorithm of a good random number generator or an implementation thereof in a higher-level language — like COBOL, for instance. I'd sure rather get one ready-made than have to take the time to learn the theory necessary to develop one myself!

Ed. Note: On receipt of the letter above, I replied to Tom encouraging him to pursue the idea further and possibly work out an example or two. A few weeks later, I received his reply.

I've been doing considerable thinking since I wrote you last about the problem of expressing algorithms in a way that would be appropriate for *Creative Computing.* I've worked out an approach that feels pretty good to me; I'd be very interested in your reactions.

My approach grew out of the thought that, even tho there are segments of the *Creative Computing* readership that do not wish to write programs in BASIC, it is none-the-less probably true that close to 100% of the readers can read and understand BASIC. Therefore, why not use a kind of "pseudo-BASIC" as an algorithm language?

After some experimentation, I've roughed out some of the properties and guidelines for a "psuedo-BASIC" algorithm language. First, a major difference: *no* line numbers in pseudo-BASIC. Instead, I've gone to a section/paragraph structure very much like COBOL. This allows one to segment the logic blocks of an algorithm quite nicely and to name them in a very meaningful way. The scope of sections or paragraphs is just as in COBOL: the scope extends only to the next occurring section or paragraph name — no nesting of scopes. Statements which would normally take line numbers in their syntex (such as GO TO, IF... THEN..., and GOSUB) now take paragraph names.

Secondly, I've confined my use of BASIC to a set of "care" statements—no extended or generalized statements. My feeling here is that clarity and immediate understandability are more important than economy of expression.

I've also tried, insofar as it is reasonable, to adhere to the spirit of structured programming: no superfluous GOTOs or multiple exits from a sub-process. Finally, I've tried to make consistant use of indenting in the writing of an algorithm, to exhibit the nesting and sub-process structure of the algorithm. I've also gotten in the habit of using a non-standard variable "REPLY" for YES/NO type responses from a sub-process or the TTY—like a kind of special flip/flop arrangement.

Rather than try to talk this thing to death, I've whipped up a few examples of the application of these ideas to some programs from previous issues of *Creative Computing.*

This second example, an algorithm for NOT ONE, exhibits the power of an algorithmic approach. I've written the mainline algorithm (MAINLINE SECTION) in a rather general way. The number of rounds in a game is left open; the number of dice rolled and the number of sides on a die is left open; the identity of the two players is not assumed in the structure of the main algorithm. All of these game parameters are nailed down in the ALGORITHM-UTILITIES SECTION. Thus, the utilities section takes the form of "pluggable" modules which I can alter easily. I can, for instance, investigate the effect on the game of making PLAYER-1 the computer and PLAYER-2 the human—I suspect the human's strategy would change under these conditions. I could give the algorithm to a student in an introductory course in probability and ask him to supply the data array for the INITIALIZE-PLAYER-2-STRATEGY; an obvious test of his understanding would be to let someone else attempt to beat his program. All sorts of other approaches to a strategy leap to mind; to test them, we set up the program with your strategy as an automated Player 1 and my strategy as an automated Player 2, and let the program run off a few hundred games — whose strategy is best? What does "best" mean: short range success, long range success, or something else? This is the kind chain-reaction investigation that so excites me about working with algorithms instead of a specific program. So many possibilities leap right out at you from a well-designed algorithm.

In any event, that's where I have gotten to in the last few weeks. As I mentioned earlier, it feels pretty good to me, but it's difficult to be completely objective about one's own ideas; I'm looking forward eagerly to your reactions. There are a couple of aspects of my pseudo-BASIC that feel rough to me yet. One is the naming of variables— BASIC's convention is a little restrictive and, I think, tends to obscure the usage of a variable. The other area that has me really stumped right now is string handling and character manipulation. As a *programming* technique, BASIC's string variable machinery may be adequate (tho I personally have never felt comfortable with that approach) but the same machinery (at the language level, of course) for the statement of algorithms doesn't please me at all.

By the by, before I forget: you may indeed consider me a volunteer to extract algorithms from any programs you might care to send along. In this regard, I'm already working on an algorithm for LUNAR LANDER and all of the related programs. It'll be a while on that one, tho, because I want to make sure I'm on firm ground with my physical model first. Also, don't drop STAR TREK on me right away — I'm not sure I'm ready for a trauma of that magnitude just yet! Otherwise, let 'er rip and we'll see what happens.

If I can arrive at some sort of reasonable way to handle string variables in pseudo-BASIC, there are a couple of programs in my file that I'd like to write up for publication. I wrote them initially a few years ago, but they need some cleaning-up and I don't have access to BASIC now so I'd have to develop them as algorithms. One is a program that generates "plots" for those horrible Japanese Sci-Fi movies that you see on TV in the wee hours of the morning. The other program came out of an interesting course in computational linguistics I took a few years ago. The program takes Spanish words and separates them into syllables. Turns out that syllabification in Spanish is a very regular, well-defined process.

I think I've about run out of gas for now; looking forward to your reactions.

Readers: it's up to you.

Example #1 — GUESS A NUMBER

"Guess a Number" is a program in which the computer chooses a number and the user tries to guess it in as few tries as possible. The computer tells the user after each guess whether it was high or low.

Versions of the program have appeared in *Creative Computing,* Sep/Oct 1975 and Mar/Apr 1977 (see "A Musical Number Guessing Game" elsewhere in this issue), *The Best of Creative Computing — Vol. 1, PCC,* and *What To Do After You Hit Return.*

```
MAINLINE SECTION.
  START-HERE.
        REM *** PRINT INSTRUCTIONS FOR GAME.
        GOSUB PRINT-INSTRUCTIONS
GET-COMPUTER-NUMBER.
        REM *** COMPUTER "THINKS" OF A NUMBER,
        X
        LET X= INT (100 # RND (  )) + 1
        PRINT "OK, I HAVE A NUMBER. START GUESS-
        ING."
  HUMAN-GUESS.
        REM *** HUMAN MAKES A GUESS, G
        PRINT "WHAT IS YOUR GUESS?"
        INPUT G
  CHECK-GUESS.
        REM *** COMPARE GUESS TO TARGET
        IF G=X THEN GUESS-IS-CORRECT
        IF G  V  X THEN GUESS-IS-TOO-BIG
        GO TO GUESS-IS-TOO-SMALL
  GUESS-IS-TOO-SMALL.
        PRINT "TOO SMALL, TRY A LARGER NUMBER."
        GO TO HUMAN-GUESS
  GUESS-IS-TOO-BIG.
        PRINT "TOO BIG, TRY A SMALLER NUMBER."
        GO TO HUMAN-GUESS
  GUESS-IS-CORRECT.
        PRINT "YOU GUESSED IT! LET'S PLAY AGAIN."
        GO TO GET-COMPUTER-NUMBER
PROGRAM-UTILITIES SECTION.
  PRINT-INSTRUCTIONS
        PRINT "HERE ARE PLAYING INSTRUCTIONS:"
          •
          •
          •
        RETURN
```

© CREATIVE COMPUTING

"It makes everything easy as 3.1415926 ..."

Example #2 — NOT ONE

"Not One" was originally presented in *Creative Computing*, Nov/Dec 1974 and Mar/Apr 1975, but is now available in *The Best of Creative Computing — Vol. 1* (pp 252-253).

The game consists of 10 rounds. On each turn a player rolls a pair of dice from 1 to N times. His score is the total of all rolls, however, if any roll equals the first roll on that turn, the turn ends with a score of 0. A more complete description can be found in the above references.

```
MAINLINE SECTION.
 BEGIN-GAME.
       REM *** INITIALIZE PLAYER'S TOTAL SCORES.
       LET P1 = 0
       LET P2 = 0
 START-ROUND.
       REM *** A ROUND CONSISTS OF 1 TURN FOR
       REM *** EACH PLAYER; THERE ARE R1 ROUNDS
       REM *** IN A GAME
       FOR R = 1 to R1
       PRINT "ROUND"; R
 PLAYER-1-TURN.
       REM *** PLAYER'S FIRST ROLL ESTABLISHES
       HIS
       REM *** TARGET NUMBER; PLAYER CRAPS OUT
       REM *** IF ANY SUBSEQUENT ROLL HITS THE
       TARGET.
       GOSUB INITIALIZE-PLAYER-1-STRATEGY
 PLAYER-1-FIRST-ROLL.
       REM *** ESTABLISH PLAYER 1 TARGET
       NUMBER.
       REM *** TOTAL OF ROLL IS RETURNED IN T.
       REM *** NUMBER OF DICE ROLLED IS IM-
       MATERIAL
       REM *** AT THIS LEVEL OF THE LOGIC.
       GOSUB ROLL-AND-TOTAL-DICE
       LET T1 = T
       LET S1 = T
 PLAYER-1-SHOW-AND-TELL.
       REM *** PLAYER 1 IS TOLD THE RESULT OF
       REM *** THE ROLL AND ASKED WHAT HE
       WANTS
       REM *** TO DO NEXT.
       GOSUB TELL-PLAYER-1-THE-RESULT
       GOSUB ASK-PLAYER-1-WHAT-NEXT
       IF REPLY = "STOP" THEN PLAYER-1-TURN-ENDS
 PLAYER-1-ROLLS-AGAIN.
       REM *** PLAYER 1 HAS ELECTED TO CONTINUE
       GOSUB ROLL-AND-TOTAL-DICE
       IF T=T1 THEN PLAYER-1-CRAPS-OUT
       LET S1 = S1 + T
       GO TO PLAYER-1-SHOW-AND-TELL
 PLAYER-1-CRAPS-OUT.
       REM *** PLAYER 1 CRAPPED OUT; HIS SCORE
       REM *** FOR ROUND IS 0.
       LET S1 = 0
       GOSUB TELL-PLAYER-1-THE-RESULT
       GOSUB TELL-PLAYER-1-BAD-NEWS
 PLAYER-1-TURN-ENDS.
       REM *** PLAYER 1 HAS COMPLETED HIS TURN;
       REM *** UPDATE HIS SCORE.
       LET P1 = P1 + S1
 PLAYER-2-TURN
       REM *** THE ALGORITHM FOR PLAYER 2's
       TURN
       REM *** IS THE SAME AS FOR PLAYER 1.
       GOSUB INITIALIZE-PLAYER-2-STRATEGY
 PLAYER-2-FIRST-ROLL.
       GOSUB ROLL-AND-TOTAL DICE
       LET T2 = T
       LET S2 = T
 PLAYER-2-SHOW-AND-TELL
       GOSUB TELL-PLAYER-2-THE-RESULT
       GOSUB ASK-PLAYER-2-WHAT-NEXT
       IF REPLY ="STOP" THEN PLAYER-2-TURN-ENDS
 PLAYER-2-ROLLS-AGAIN.
       GOSUB ROLL-AND-TOTAL-DICE
       IF T=T2 THEN PLAYER-2-CRAPS-OUT
       LET S2=S2+T
       GO TO PLAYER-2-SHOW-AND-TELL
 PLAYER-2-CRAPS-OUT.
       LET S2=0
       GOSUB TELL-PLAYER-2-THE RESULT
       GOSUB TELL-PLAYER-2-BAD-NEWS
 PLAYER-2-TURN-ENDS.
       LET P2 = P2 +S2
 END-ROUND.
       GOSUB PRINT-SCORE-FOR-ROUND
       NEXT R
 END-OF-GAME.
       GOSUB PRINT-GAME-SCORE
       STOP

ALGORITHM-UTILITIES SECTION.
 INITIALIZE-PLAYER-1-STRATEGY.
       REM *** LET PLAYER 1 INITIALIZE HIS OWN
       STRATEGY
       RETURN
       ROLL-AND-TOTAL-DICE.
       REM *** ROLL 2 DICE, PASS TOTAL THROUGH
       VARIABLE T.
       LET D1 = INT (6 * RND ( ) ) + 1
       LET D2 = INT (6 * RND ( ) )+ 1
       LET T = D1 + D2
       RETURN
 TELL-PLAYER-1-THE-RESULT
       PRINT T
       RETURN
 ASK-PLAYER-1-WHAT-NEXT.
       PRINT "STOP OR ROLL AGAIN?"
       INPUT REPLY
       RETURN
 TELL-PLAYER-1-BAD-NEWS.
       PRINT "YOU GET A ZERO FOR THIS ROUND."
       RETURN
 INITIALIZE-PLAYER-2-STRATEGY.
       REM *** LOAD PROBABILITY VECTOR
       RESTORE
       FOR I = 2 TO 12
       READ Z(I)
       NEXT I
       RETURN
 TELL-PLAYER-2-THE-RESULT.
       RETURN
 ASK-PLAYER-2-WHAT-NEXT.
       LET Z (T2) = Z (T2) - 1.
       IF Z (T2) 1 THEN PLAYER-2-STOPS
       PLAYER-2-GOES-AGAIN.
       LET REPLY = "GO"
       GO TO ASK-PLAYER-2-EXIT
       PLAYER-2-STOPS.
       LET REPLY = "STOP"
       ASK-PLAYER-2-EXIT.
       RETURN
 TELL-PLAYER-2-BAD-NEWS.
       RETURN
```

Creative Programming Techniques. . . .

Shuffling

by John Jaworski
Hatfield Polytechnic, England

When faced with the problem of printing out the integers from 1 to 10 in a random order (without repetitions), the following program is an excellent example of how not to proceed:

```
100  FOR I = 1 TO 10
110  N = INT(10*RND+1)
120  PRINT N;
130  NEXT I
140  END
```

As we are taking no precautions against repetitions, we can be almost certain to get some. In fact, if my rapid calculations are correct, the probability of getting what we are looking for is minute: 3.6 x 10^{-4} – rather less than 4 correct solutions in 10,000 trials!

Somewhat better, on the face of it, is the next example. Here we provide an array M, choose a random integer and only insert it into the array M when we have checked that this integer is not already present.

Note that statment 100 is superfluous in the BASIC language, but it is as well to remind ourselves of the nature of M.

```
100  DIM M(10)
110  K=1
120  N=INT(10*RND+1)
130  FOR J=1 TO K
140  IF M(J)=N THEN 120
150  NEXT J
160  M(K)=N
170  K=K+1
180  IF K<11 THEN 120
190  MAT PRINT M;
200  END
```

Unfortunately, while producing results, this is not at all economical on time. When the array is almost full, say with 9 numbers inserted, there is in fact no choice at all for the last element, but only one chance in ten that statement 120 will select the correct integer for us – 90% of the work done by the program at this stage is wasted. This of course is more acute when shuffling rather more integers!

One rather more elegant method is the following: choose ten random *real* numbers less than 1, using RND. Associate these with the ten integers 1 - 10 and then sort them into order, shifting the integers around at the same time:

e.g.

BEFORE		*AFTER*	
1	0.1143	0.0954	9
2	0.9317	0.1143	1
3	0.5120	0.2671	5
4	0.3367	0.2758	7
5	0.2671	0.3154	10
6	0.8815	0.3367	4
7	0.2758	0.4186	8
8	0.4186	0.5120	3
9	0.0954	0.8815	6
10	0.3154	0.9317	2

We can sort the real numbers by any sorting method we have found suitable. Here is a demonstration program that uses the 'ripple' sort (which you may know under some other name).

```
100  DIM A(10),P(10)          A contains the random
110  FOR I = 1 TO 10          numbers and P the inte-
120  P(I) = I                 gers that will eventually
130  A(I) = RND               be printed. This loop
140  NEXT I                   sets up these arrays.

150  L = 10                   The limit on compari-
160  F = 0                    sons, L is set to 10, the
170  I = 1                    flag is set and we begin
                              our comparisons at the
                              beginning of the list.

180  IF A(I) > A(I+1) THEN 260  compare adjacent
                                numbers.

190  T = A(I)                 if not in order, swap
200  A(I) = A(I+1)            A's . . .
210  A(I+1) = T

220  T = P(I)
230  P(I) = P(I+1)            . . . swap P's
240  P(I+1) = T

250  F = 1                    and set flag

260  I = I+1
270  IF I < L THEN 180        increment and branch back

280  IF F = 0 THEN 310        if in order, stop

290  L = L-1
300  GO TO 160
310  MAT PRINT P;
340  END
```

The last example underlines the fact that the more efficient and more elegant program is not always the simplest when written out.

ACHTUNG!

Alles Lookenspeepers

DAS COMPUTENMACHINE IS NICHT FUR GEFINGERPOKEN UND MITTENGRABBEN. IST EASY SCHNAPPEN DER SPRINGENWERK, BLOWENFUSEN, UND POPPENCORKEN MIT SPITZENSPARKEN.

IST NICHT FUR GEWERKEN BY DAS DUMMKOPFEN. DAS RUBBERNECKEN SIGHTSEEREN KEEPEN HANDS IN DAS POCKETS–RELAXEN UND WATCH DAS BLINKENLIGHTS.

Creative Programming Techniques. . . .

A Crooked Shuffle

A Case Study in Bebugging The Programmer

Alan Filipski*

In an article on shuffling in the Jan.-Feb. 1977 issue of Creative Computing, John Jaworski considered the problem of generating the numbers from 1 to N in a random order without repetition (a "random permutation"). Both solutions given in that article have execution times on the order of N^2, *i.e.* shuffling 10N items would take about 100 times as long as shuffling N items for large N. My first reaction was that there is an obvious way to shuffle in linear time (time proportional to N for large N). It turns out that there is indeed such a way, but we have to be a little careful about what is "obvious." The following account traces the development of such an algorithm, pointing out some tempting fallacies along the way.

The germ of the idea is this: We first create an array containing the numbers from 1 to N in order. We then proceed to destroy that order by interchanging the contents of each location in turn with the contents of a location selected in some random fashion. To make this idea more precise, we could say

1. Generate an array A containing the numbers from 1 to N in order.

2. For each i from 1 to N:
   Pick a random integer j between 1 and N and switch A_i with A_j.

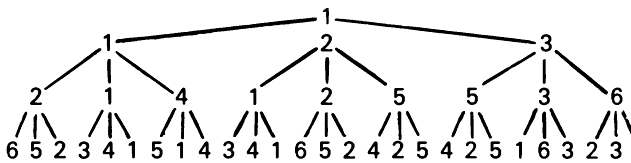
Thus every item gets switched at least once and on the average twice. This would be easy to program and takes linear time to execute. The method obviously mixes things up so thoroughly that we certainly must be getting random permutations. Of course, we could prove it if we wanted to, but proofs are just pedantic exercises, and besides, we have programming to do, right? Well, just for laughs, let's try to prove that this algorithm does what we want.

First, we should clarify exactly what we mean by the phrase "generating the numbers from 1 to N in a random order without repetition." The "without repetition" criterion is easy to verify because it is a property which must apply to *each* sequence generated. The "random order" criterion requires a little more thought, since it is a notion which applies to the entire class of permutations generated, but not to any single permutation (at least not without arousing some statistical and philosophical demons who are better left undisturbed). As a definition of "random order" we might venture to say that the probability of the number I appearing in the J^{th} position should be 1/N for all I and J between 1 and N. This insures that any number has an equal chance of appearing anywhere, so the program which satisfies this criterion must be generating all permutations at random, right? Wrong. Suppose N = 3. Then the possible permutations are:

$P_1 = (1\ 2\ 3)$ $P_2 = (2\ 1\ 3)$ $P_3 = (3\ 2\ 1)$
$P_4 = (1\ 3\ 2)$ $P_5 = (2\ 3\ 1)$ $P_6 = (3\ 1\ 2)$

Consider a program which outputs P_1 or P_5 or P_6, each with probability 1/3. This satisfies our proposed criterion, but is obviously not what we mean by a random shuffle, because the probability of generaging P_2, P_3, or P_4 is zero. This suggests that what we really want to say is that our program must generate any permutation with equal probability (probability 1/n! in fact, since there are N! different permutations.) Now that we know what we want, let's see how our program goes about producing it.

Consider the case when N = 3. The program starts with P_1. The first interchange transforms it to either P_1, P_2, or P_3 with probability 1/3 each. Two more interchanges are then performed on the result giving the final permutation. Since we have three choices at each of the stages, there is a total of 27 equally likely series of interchanges. Of course, some sequences of interchanges must produce the same result since only six different permutations are possible. We can represent these successive transformations by a tree as follows:



We note that at the final level, P_1, P_3, and P_6 occur four times each, while P_2, P_4, and P_5 occur five times each. The latter are therefore more likely to be generated than the former. Of course, if we were smart, we could have foreseen trouble just be observing that 6 does not divide 27 evenly.

So it appears that our algorithm does a rather slipshod shuffle. Well, what now? Is the idea bankrupt? Maybe noit. Consider

---

**Department of Mathematics, Central Michigan University, Mt. Pleasant, MI 48859.*

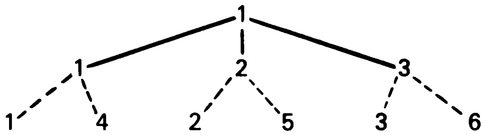
a modification of the technique: We start with the array A_1, A_2, A_3, . . . A_N containing the numbers from 1 through N in order. We begin as before by interchanging A_1 with the contents of a randomly selected location. We now want to set A_2 equal to one of the *remaining* items. This is the key to the rehabilitation of the algorithm. We accomplish this by selecting a random integer J between 2 and N and interchanging the contents of A_2 with A_J. Continuing in this way, our algorithm now becomes:

1. Generate an array A containing the numbers from 1 to N in order.

2. For each i from 1 to N: Pick a random integer j between i and N and switch A_i with A_j.

If we now display the situation for N = 3 in terms of our treee,

If we now display the situation for N = 3 in terms of our tree, we have:



which is exactly what we want, generating each permutation with probability 1/n!. We can now implement the algorithm with the following program:

```
100 DIM M(52)
110 LET N = 52
120 FOR I = 1 TO N
130 LET M(I) = I
140 NEXT I
150 FOR I = 1 TO N-1
160 LET J = INT(RND(0)*(N-I + 1)) + I
170 LET T = M(I)
180 LET M(I) = M(J)
190 LET M(J) = T
200 NEXT I
210 MAT PRINT M;
220 END
```

Thus we arrive at an efficient and simple solution to the original problem. As you may have guessed, however, the point of this paper is not the presentation of a shuffling algorithm which works in linear time (which can be found, for example, in Knuth's *Seminumerical Algorithms*) but rather an illustration of potential traps along the path of algorithm development. If you had (as I did) a tendency to swallow the argument that the first version of the algorithm "mixes things up so thoroughly that we must be getting random permutations," you have a bug in your quantitative intuition. This sort of bug is more insidious than any program bug since it potentially affects any algorithm you might develop. The existence of such bugs is not often publicized since it is ever the wont of mathematicians to display their creations in the austere beauty of their perfected form and to be ashamed of the false starts and jumpted conclusions along the way. (The exception here is the "paradox" which is such a dramatic and epidemic bug that it has entertainment value.)

If we are to make progress in exorcising these bugs, it behooves us to stop at least and recognize them for what they are. In the future, would we be more suspicious of a line like the "mix-em-up" argument? Is it clear that the picture of the tree leads to a proof in the case of the revised algorithm? Is it reasonable that N = 3 should yield a sufficiently general example to discredit the first algorithm, but that N = 2 should not? The consideration of such questions would be a first step in the debugging of the programmer. ■

Shuffling Revisited

The article on "Shuffling," in the Jan-Feb issue (page 77) drew a large response from readers who offered shorter or "more elegant" ways of solving the problem. Here are a few of the letters:

"More Elegant"

Dear Editor:

John Jaworski's article, "SHUFFLING", in the January-February issue contains a minor error in statement 180. The > symbol will produce a descending sort rather than the ascending sort shown in the before-and-after example. This has no real effect on the outcome except to reverse the order of the randomized integers.

Shown below are two routines which are more elegant than the shuffling technique (from the standpoint of requiring less iterations for a typical run and being more concise in code length):

The first uses a search technique borrowed from hashing algorithms rather than performing a sort.

```
100 DIM A(10), P(10)
110 FOR I = 1 TO 10
120 A(I) = I
130 NEXT I
```

A contains a table of integers, P will contain the integers in random sequence. The first loop puts the integers in A.

```
140 FOR I = 1 TO 10
150 J = INT (10*RND + 1)
160 IF A(J) > 0 THEN 210
170 J = J + 1
180 IF J < 11 THEN 160
190 J = 1
200 GO TO 160
```

Generate a random integer use this integer to access A Scan through A until you find an integer which has not been used yet.

```
210 P(I) = A(J)
220 A(J) = 0
230 NEXT I
```

Place the next integer in the output table and remove this integer from A

```
240 MAT PRINT P
250 END
```

Print P when all integers are moved.

The second routine shows how this same function appears in APL:

David D. Keefe
Tillson, NY

"Each Loop Used Only Once"

Dear Editor:

On reading "Shuffling" by Jaworski in Creative Programming Techniques, January-February 1977 issue, I notice a sort is required. For longer lists, this can be a time-consuming routine. Here is a routine to shuffle 52 cards in one pass. Cards are picked one at a time and each of the remaining cards has an equal chance of being picked.

```
100 DIM M(52)
110 N = 52
120 FOR I = 1 TO N
130 M(I) = I
140 NEXT I
```

Enter numbers 1 to N in list in order.

```
150 FOR I = 1 to N-1
160 R = (N + 1-I)*RND(1)
170 R = INT(R) + I
180 T = M(R)
190 M(R) = M(I)
200 M(I) = T
210 NEXT I
```

Pick number R between I and N.

Exchange entries I and R.

Each loop is used only once.

James Murphy
Associate Professor
California State College,
San Bernadino, CA 92407

"Simpler and Smaller"

Dear Editor:

The article by John Jaworski on "Shuffling" was very interesting. However, I am unimpressed by the little "moral" at the end. Several years ago I constructed a card-shuffling program based on an explanation of permutation theory based on a mail-clerk and pigeon holes. I don't remember the source of the explanation or its precise details, but I do remember the algorithm. Translated to BASIC it looks something like this:

```
DIM M(10)
FOR I  =  1 TO 10              Initialize the array—this step
M(I)  =  I                     is only required once and the
NEXT I                         program can be used to
                               generate as many permuta-
                               tions as you wish.

FOR J  =  1 to 9
K  =  M(J)
L = INT ((11-J)*RND + 1)
M(J)  =  M(L + J - 1)
M(L)  =  K
NEXT J
```

As you can see, the algorithm chooses each element of the permutation randomly from the numbers not previously chosen. The advantages over sorting are: (1) less memory is required (only one vector instead of 2), (2) fewer exchanges per permutation (no sorting program can beat N - 1 consistently), (3) no comparisons at all and (4) the program itself is much simpler and smaller.

The January February issue was my first experience of your magazine—I enjoyed it thoroughly! Keep on computing!

Dean Ritchie
Systems Programming Manager
Computing Center
Washington State University
Pullman, WA 99163

"Requires Less Memory and Time"

Dear Editor:

This letter could be headed "A Better Way to Shuffle." I was disappointed to see that John Jaworski omitted one easy shuffling technique—random indexing—from his treatment of BASIC programming, and wish to fill the void. To shuffle an array using random indexing is to choose elements by using random numbers to calculate addresses. The following BASIC statement will calculate the address of one of an N-element array with subscripts ranging from 1 to N. If your BASIC interpreter recognizes the zeroth element of an array, then the statement will have to be changed to avoid wasting an array element.

```
I  =  INT(N*RND(0) + 1)
```

After the Ith element is removed from the array and stored in a safe location, the array is packed by moving the top elements down one space, and N is decremented by 1. Another element is selected using the same method, and the process repeated until the array is used up. You might think two large arrays would be needed, one to hold the source array of elements, and one to hold the shuffled array, but that isn't so. Remember that after the Ith element was selected, the remaining elements were packed together to eliminate the gap. That left a gap at the top of the array where the element would fit nicely. Packing the array isn't difficult, either. Because the shuffled array is supposed to be in random sequence, it really doesn't matter what order the source array is in. To pack the array, remove the unselected upper element from the top of the array and plug the gap. Putting it all together for a program to print nine digit numbers, with no two digits the same, yields the following BASIC code:

```
 100 DIM A(9)
 200 REM FILL THE ARRAY WITH
 300 REM THE DIGITS FROM 1
 400 REM TO 9
 500 FOR I  =  1 TO 9
 600 LET A(I)  =  I
 700 NEXT I
 800 REM THE SHUFFLING ROUTINE
 900 FOR I  =  9 TO 2 STEP  - 1
1000 LET J  =  INT(I*RND(0) + 1)
1100 IF J>I THEN 1000
1200 LET T  =  A(J)
1300 LET A(J)  =  A(I)
1400 LET A(I)  =  T
1500 NEXT I
1600 FOR I  =  1 TO 9
1700 PRINT A(I)
1800 NEXT I
1900 END
```

This program requires less memory and time than the routines provided by Mr. Jaworski. Speed and space-saving are important, especially in a program like BLACKJACK which shuffles a 52-card deck several times.

William R. Hamblen
946 Evans Rd.
Nashville, TN 37204

"At Random"

Dear Editor:

While looking through the January/February *Creative Computing*, I noticed the "Shuffling" article (J. Jaworski, p.77), thought, "There, but for the grace of Iverson, goes 10?10," and turned the page. But then, upon a closer reading of the magazine, I discovered the same technique advocated on the very facing page! And with the same ineluctable bubble sort! This was too much. Even with a good sort, the program is inefficient. The obvious way to shuffle 10 or any number of n numbers is: a) pick one at random b) pick one of those remaining c) continue until none are left. Since the two sets, picked and unpicked, will always total 10 (or however many) numbers, we just move the boundary through the array, exchanging the number whose place we want with the one we wish to put there. BASICly:

```
100 DIM A(10)                          A is 1, 2, . . . ,10.
110 FOR I = 1 TO 10
120 A(I) = I
130 NEXT I
140 FOR I = 1 TO 9                      I is the boundary.
150 K = I + INT (RND*[11-I])          K is a random number
160 T = A(I)                                 from I to 10.
170 A(I) = A(K)                                  Exchange
180 A(K) = T
190 NEXT I
200 MAT PRINT A;                                    Done.
210 END
```

Using the sorting method squares the time (depending on the sort) and doubles the space (code and arrays) that the program requires.

J. Storrs Hall
New Brunswick, NJ

"Faster"

Dear Editor:

I read the article in January/February *Creative Computing* on Shuffling numbers.

I have a program which also shuffles numbers, which is simpler to program and executes faster than the program in the article.

I want to share it with your readers.

```
10 RANDOMIZE
100 DIM A(10, P(10)
110 FOR I = 1 TO 10
120 LET A(I) = I
130 NEXT I
140 FOR I = 1 TO 10
150 LET T = INT((11-I)*RND) + 1
160 LET P(I) = A(T)
170 LET A(T) = A(11-I)
180NEXT I
190 MAT PRINT P;
200 END
```

Elliott Werner
ARCDATA Systems
66-51 Booth Street
Rego Park, NY 11374

GRID ADDRESSING

by

Gerard Akkerhuis

US Dependent Schools, European Area

APO New York 09175

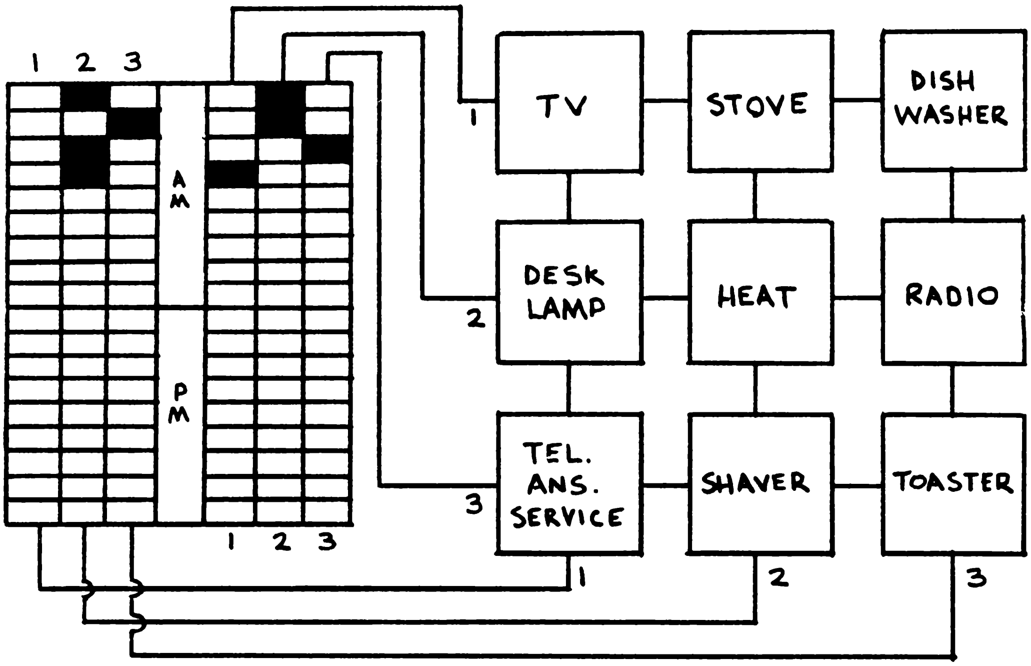
Grid addressing is a technique which has many applications in contemporary technology. An understanding of this method erases part of the mystery which surrounds the computer's instantaneous manipulation of both instructions and data through only data.

A fictitious kitchen grid will be used to illustrate the concept of grid addressing. A bachelor enjoyed waking to a warm room filled with soft music. He would rise, shave, have eggs, coffee, toast, and run the dishwasher. Then he would read the paper and watch the TV Morning Show. When he left the house for work, his answering service monitored his telephone calls.

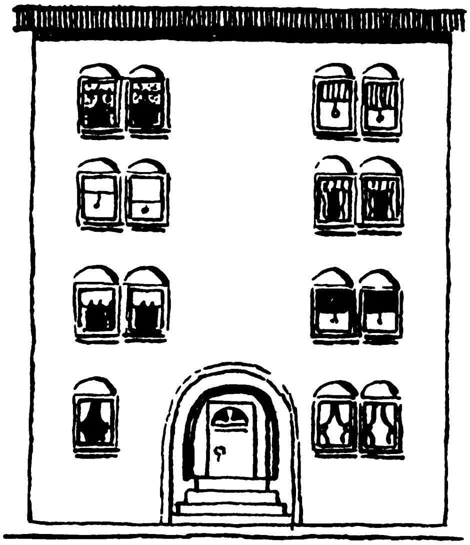
After work the bachelor checked his answering service and listened to music while he shaved. Then he ate, ran the dishwasher, read the evening paper, watched the late show, had toast and milk, and went to bed with the heat turned down.

If the bachelor's appliance grid is that shown in the figure and if half of the electricity required to operate an appliance went up the vertical line and half across the horizontal line, what numbers would he give his grid for his morning program and what numbers would he give his grid for his evening program? We've filled in the first four morning operations in the grid (memory). Can you fill in the rest?

When I was in Europe at the 2nd World Computers in Education Conference in September 1975, Sam Calvin and Gerard Akkerhuis were kind enough to give Sandy and I lodging and "guide service" in the Frankfurt area. Gerard also gave me a wealth of outstanding material that he wrote for use in the USDESEA schools. Unfortunately, much of it is geared to the Interdata 7/16, a not-very-common computer in schools and homes. However, as we get the material generalized, it will appear on these pages (eventually). — DHA



J, J, J & T!



(1) Joe, Jack, John and Tom live one on each floor of a four-story apartment.

(2) Their ages are 10, 9, 8, 5, but not necessarily in that order.

(3) Joe lives directly above the 9-year-old and directly below the 8-year-old.

(4) Jack has to pass by the 5-year-old to leave the building from his apartment.

(5) Jack is more than one floor away from Tom, who is more than one year younger than Jack.

Find the ages and on which floor each of the boys lives.

Find the Presidents and Vice-Presidents of the United States

Names of the first 37 Presidents and 30 Vice-Presidents of the United States can be found among these letters. The name of each is hidden either forward, backward, up, down, or diagonally. Names appearing more than once on the list appear as many times in the matrix, for example. Johnson appears three times. (One multiple name appears only once, however, which one?)

Computer programs have been written to hide names in a matrix. Can you write such a program? Using one (or by hand) can you pack the names any more efficiently than is done here (using fewer total matrix elements — square or rectangular)?

Computer programs have also been written to locate names in a matrix. If you were to write such a program, what is the best search technique? There are many and the problem is no easier than finding an efficient sorting method.

Lastly, how can the matrix here be modified to hide the name of our latest president and vice-president, CARTER and MONDALE by changing as few letters as possible?

```
R Z F H U M P H R E Y D Y E L K R A B L L A H S R A M
O S K C I R D N E H O B A R T C L R S K N A B R I A F
C T R E N R A G R E L E E H W E G D I R N I K C E R B
K E E R T R U M A N P R E W O H N E S I E C A L L A W
E V T C O L X A F L O C S E Y A H I S O N O S N H O J
F E O O O O E Y B N D L E I F R A G M R E V O O H M L
E N C W G O S V D P L R U H T R A H A R R I S O N E D
L S S L I N L E E E R O M L L I F W D N A L E V E L C
L O E H I L I I V S N L C N O S I D A M C K I N L E Y
E N C W E N S D D E O N N N B O F P D S O F P Q E W J
R A U N A R T O R G L O E O I N O R A T H N O C F I O
H G R I E D M O N A E T R K S L R I M L A I R R R L H
A N T X D G N A N A H C U B K K D X S O R E N O D S N
M E I O T R E L N Q T S B U R R C N O X I N D G E O S
L W S N N O T R O M G F N G N I K A P P R E L Y T N O
I N O S N H O J R O L Y A T N A R G J E F F E R S O N
N S N I K P M O T Y A D V T S A L L A D C A L H O U N
```

Presidents

WASHINGTON
ADAMS
JEFFERSON
MADISON
MONROE
ADAMS
JACKSON
VAN BUREN
HARRISON
TYLER
POLK
TAYLOR
FILLMORE
PIERCE
BUCHANAN
LINCOLN
JOHNSON
GRANT
HAYES
GARFIELD
ARTHUR
CLEVELAND
HARRISON
MCKINLEY
ROOSEVELT
TAFT
WILSON
HARDING
COOLIDGE
HOOVER
ROOSEVELT
TRUMAN
EISENHOWER
KENNEDY
JOHNSON
NIXON
FORD

Vice Presidents

BURR
CLINTON
GERRY
TOMPKINS
CALHOUN
JOHNSON
DALLAS
KING
BRECKINRIDGE
HAMLIN
COLFAX
WILSON
WHEELER
HENDRICKS
MORTON
STEVENSON
HOBART
FAIRBANKS
SHERMAN
MARSHALL
DAWES
CURTIS
GARNER
WALLACE
BARKLEY
NIXON
HUMPHREY
AGNEW
FORD
ROCKEFELLER

This is the first in a series of articles about strategies or approaches for solving practical problems on the computer. Readers will find the heuristics and rules of thumb discussed in these articles are independent of subject matter and of great value in solving all types of programming problems from simple to very complex. —DHA.

THINKING STRATEGIES WITH THE COMPUTER: INFERENCE

Donald T. Piele and Larry E. Wood*

> *Experience in solving problems and experience in watching other people solve problems must be the basis on which heuristic is built.*
>
> G. Polya

Some 32 years ago, in 1945, George Polya published a little book called *How To Solve It.* Judging from the title, one might expect to find inside special techniques and sure-fire algorithms that guarantee solutions to specific problems. But this is not what Polya's book is about. Instead, it is packed full of ideas and 'rules of thumb' that are useful in attacking any type of problem but do not guarantee a solution to any specific one. Polya's methods, which he labels heuristic, are derived from the experience of good problem solvers and are characterized by their generality, their independence of subject matter, and their common sense.

Inspired by the work of Polya and recent advances in the field of artificial intelligence (e.g. Newell and Simon, 1972), Wayne Wickelgren published a similar book, *How To Solve Problems* in 1974. This book contains detailed explanations of several general problem solving strategies along with puzzles and games to illustrate each strategy. Puzzles are well suited for the task because they require the same logical thinking processes as problems in any subject area but they do not require any special knowledge. Their only drawback is that people sometimes refuse to take them seriously. They fail to see any connection between the thinking skills needed to solve a frivolous puzzle and those needed to solve more practical problems.

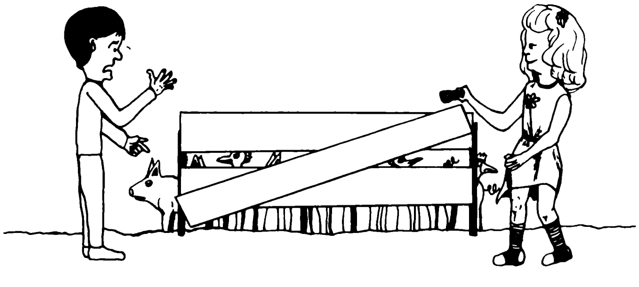
Recently, we have been studying the problem solving strategies of Polya and Wickelgren and have been extremely impressed with their generality and power. In preparation for a course on Thinking Strategies at UW-Parkside, we have collected many examples of puzzles and games from the pages of such classic works as The Moscow Puzzles, and Mathematical Puzzles of Sam Lloyd for use in practicing each strategy. Now we are exploring ways in which computer programming can be incorporated with these skills to solve even more complex problems. We would like to share some of our ideas in a series of articles for *Creative Computing* demonstrating the added power of heuristic problem solving skills when used in conjunction with the computer.

Inference

In this first article, we will discuss the strategy of *inference. Webster's New Collegiate Dictionary* defines inference as "a logical conclusion from given data or premises, a judgment derived by reasoning or implication." As a heuristic problem solving tool, inference is more broadly defined to include meanings such as explicitly stating information that is implicit in the problem, making deductions and inductions, and generating and testing hypotheses. Viewed in this expanded sense, inference becomes a basic component of most problems. Indeed, it is difficult to imagine how any problem could be solved without it. As an example, consider the following problem.

PIGS AND CHICKENS

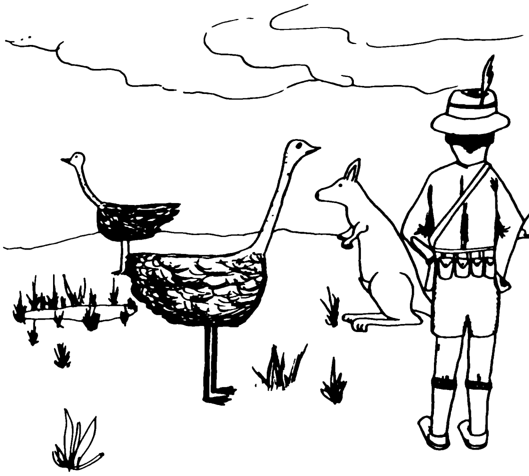
A boy and his sister visited a farm where they saw a pen filled with pigs and chickens. When they returned home, the boy observed that there were 18 animals in all, and his sister reported that she had counted a total of 50 legs. How many pigs were there in the pen?



The first step in solving any problem is to fully understand what is implied in the problem as well as what is explicitly stated. For example, in the pigs and chickens problem it is assumed one knows that pigs have four legs and chickens have only two. This may seem trivial, but if the problem were posed with kangaroos and emus, the same inference might be obvious only to an Australian problem solver. The next step is to deduce relationships that may exist between elements of the problem. For example, the total of 50 legs is equal to the number of pigs times four plus the number of chickens times two. Also, the pigs plus the chickens equals 18 animals. At this stage, anyone with a knowledge of algebra would probably symbolize the above relationships in two algebraic equations (e.g., $P+C=18$ and $4P+2C=50$) and solve for the number of pigs. This is certainly a familiar way to solve story problems, but for the purpose of emphasizing the usefulness of inference let's see how an Australian problem-solver might attack the same problem if it were posed with kangaroos and emus.

---

*University of Wisconsin-Parkside, Kenosha, Wisconsin 53140



One usually visualizes a kangaroo resting on its hind legs with its smaller front legs in the air. An emu (which resembles a large turkey) has only two legs. Thus an Australian might easily infer that with 18 kangaroos and emus, a total of 36 legs are on the ground. Since there are 50 legs in all, there must be 14 legs in the air, which belong to exactly 7 kangaroos. Such a simple solution is very unlikely to occur to someone when the problem is posed with pigs and chickens, but what is to prevent pigs from standing on their hind legs—at least in our minds? It is probably true that the less likely we are to make a particular inference, the more likely we are to label it insight. However, in this problem, it may be more appropriate to call it hind-sight!

As illustrated in the pigs and chickens problem, drawing inferences depends heavily upon prior experience. Therefore, it may be difficult to make critical inferences with complex problems or with problems from an area unfamiliar to the problem solver. To overcome this difficulty, the computer can be a very effective tool. With it, one can rapidly generate important information related to a problem, which can serve as a basis for formulating and testing hypotheses about a solution. We shall illustrate this with the following example from the field of music theory.

The Nun's Fiddle

The Greek mathematician, Pythagoras, first discovered a basic relationship between musical harmony and number. This relationship is briefly explained by Helm (1967).

> "Pluck a stretched string of any length and allow it to vibrate; it will sound a certain pitch. Allow only half of it to vibrate and the pitch will rise an octave. If two-thirds of the string vibrates, the pitch will rise a fifth above the one produced by the total length. For instance, if the total length produces C, two-thirds of the string will produce G. (The interval C-G is called a fifth because five lines and spaces on the musical staff are traversed in going from one to the other, counting C and G.) Three-fourths of the string will yield a pitch a fourth higher than the total length (F, if the total yields C) and so on. In time the fractions become more complex and the two notes represented by the resulting intervals become more dissonant if they are sounded together."

The discovery that pleasing cords correspond to exact divisions of a string by whole numbers had mystic overtones for the Pythagoreans. They inferred that if nature and number corresponded harmoniously in music it must be true that a single order, expressible in number and ratio, governed all the rhythms of nature. This led to the myth that the orbits of all heavenly bodies were related by musical intervals. "The movement of the heavens were, for them, the music of the spheres" (Bronowski, 1973). Gradually over the centuries certain ratios corresponding to musical intervals became the basis of traditional Western music. Silver (1971) explains:

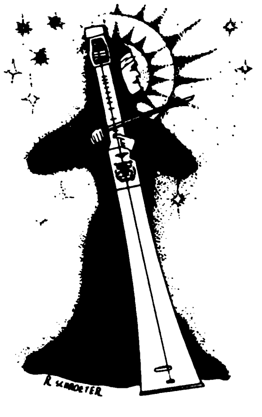
> The satisfying intervals were derived from natural harmonics, the frequencies of which are related to the natural number series 1:2:3.... Successive ratios 1:2, 2:3, 3:4...were favored. The lower ratios are pleasing; the higher ones tend to harshness and eventually become unacceptable. Certain ratios, although within the range of acceptable harshness, are regularly rejected, e.g., 6:7, 7:8, 10:11, 11:12.... There is no obvious reason for this empirical fact. However, an analysis of a large amount of material discloses that the ear prefers the following finite set called the *superparticular ratios:*

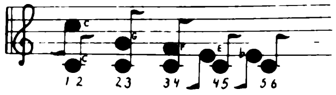
1:2 octave	8:9 major tone
2:3 perfect fifth	9:10 lesser tone
3:4 perfect fourth	15:16 diatonic semitone
4:5 major third	24:25 chromatic semitone
5:6 minor third	80:81 comma of Didymus

The relation of music to number expressed by the superparticular ratios is very beautiful, and a complete understanding of this relationship may even convince you that it is divinely inspired. The superparticular ratios are examined with the aid of the tromba marina, a late medieval bowed instrument with a single string. The instrument was frequently used by nuns and hence the German name *Nonnengeige* or nun's fiddle.

NUN'S FIDDLE

The superparticular ratios in music are related to the prime numbers and can be defined by two simple properties. Find these properties and prove that they characterize the superparticular ratios uniquely.





Length	Note	
1	C	
80/81		Comma of Didymus
24/25		Chromatic Semitone
15/16		Diatamic Semitone
9/10		Lesser Tone
8/9	D	Major Tone
5/6	E♭	Minor Third
4/5	E	Major Third
3/4	F	Perfect Fourth
2/3	G	Perfect Fifth
1/2	C	Octave

The first part of this problem can be answered by writing each whole number in a superparticular ratio in terms of its prime factors, i.e. 4= ^{2}, 6=2·3, 8=2^3, 9=3^2, 10=2·5, 15=3·5, 16=2^4, 24=2^3·3, 25=5^2, 80=2^4·5, 81=3^4. One can infer that two properties characterize these ratios: (1) Each number is of the form $2^a3^b5^c$ where a,b,c 0, and (2) the numbers in each ratio differ by one. The difficult question is whether these two properties determine the superparticular ratios uniquely. Expressed another way, is it true that if a ratio of two whole numbers satisfies conditions (1) and (2) then it must be one of the 10 superparticular ratios? This statement is, in fact, true, and it was first proved by Stormer (1897). More recently it was re-examined by Halsey and Hewitt (1972). However, to understand the formal proof requires a considerable amount of mathematical expertize. In contrast, it is quite easy to write a computer program to generate successive numbers of the form $2^a3^b5^c$ from which a number of inferences can be made. Of course these inferences do not represent a strict proof but at least they increase one's understanding of the problem to the point where a proof may be easier to discover.

Program FIDDLE was written to do precisely that—fiddle around. It allows one to specify a set of primes p_1, p_2 ... p_k from which consecutive whole numbers are generated which have these primes as their only factors. In the sample run the primes 2,3,5 are specified, and all numbers, up to 1000, which have these primes as their only factors and which differ by one are printed out. What we observe are precisely the numbers in the superparticular ratios.

Conclusion

Our ability to make inferences in problem solving is strongly dependent upon our past experience as illustrated in the pigs and chickens problem. We can overcome this difficulty in many instances by using the computer to generate information to enrich our understanding of a variety of problems—even those which are not 'divinely inspired.'

Post Script

What are the superparticular ratios for the primes 2,3,5 and 7? The answer may surprise you! Be patient; there are 23 ratios. Also, what can you infer about superparticular ratios relative to any set of primes that does not contain 2? Perhaps we have not gone far enough, and there exists two consecutive numbers beyond 1000 of the form $2^a3^b5^c$. Because a computer is limited to calculating a finite number of cases, it is impossible to absolutely rule this out. However, one can generate more evidence to weaken the case by observing the sequence of differences between successive numbers that are of the form $2^a3^b5^c$. This information is also shown in the sample run. Although the differences are not constantly increasing, at least they appear to be moving toward higher and higher values. This information prompted the authors to conjecture that for any specified distance d, successive terms of the form $2^a3^b5^c$ will eventually all differ by at least d. Shortly thereafter we found it had been proved mathematically by Stormer (1898) for any number of specified primes.

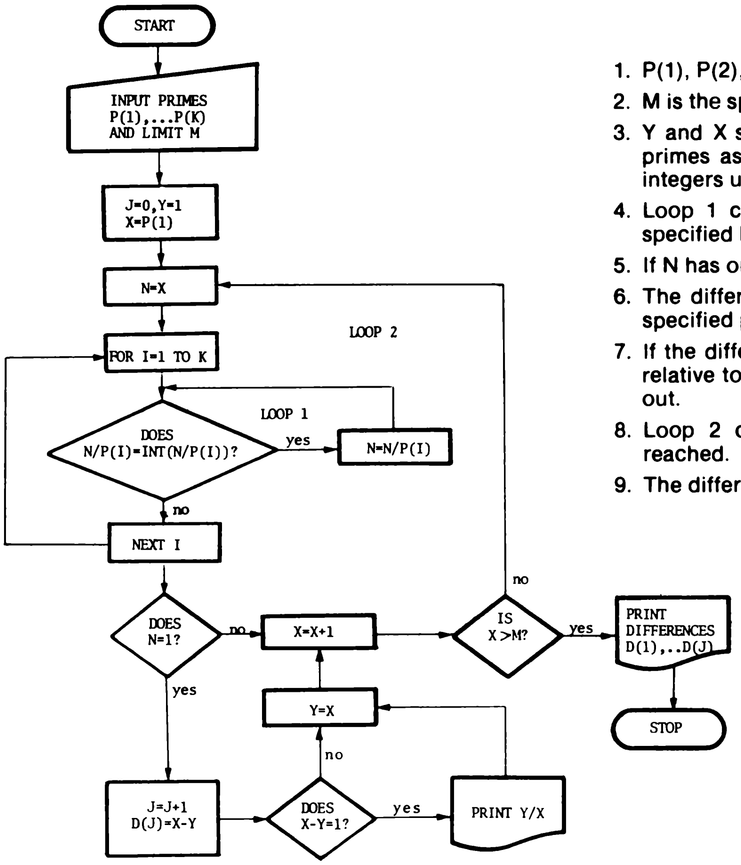
FIDDLE PROGRAM

```
LIST
FIDDLE

10  PRINT "THIS PROGRAM CAN BE USED TO STUDY SUPERPARTICULAR"
20  PRINT "RATIOS RELATIVE TO ANY SPECIFIED SET OF PRIMES."
30  PRINT
40  PRINT "  HOW MANY PRIMES DO YOU WANT TO SPECIFY";
50  INPUT K
60  PRINT "WHICH ONES ARE THEY (SMALLEST ONE FIRST)";
70  MAT  INPUT P[K]
80  PRINT "           HOW HIGH DO YOU WANT TO SEARCH ";
90  INPUT M
100  PRINT
110  PRINT "THE SUPERPARTICULAR RATIOS UP TO";M
120  PRINT "FOR THE SPECIFIED PRIMES ARE:"
130  PRINT
140  DIM D[1000]
150  J=0
160  Y=1
170  X=P[1]
180  N=X
190  FOR I=1 TO K
200  IF N/P[I]#INT(N/P[I]) THEN 230
210  N=N/P[I]
220  GOTO 200
230  NEXT I
240  IF N#1 THEN 300
250  J=J+1
260  D[J]=X-Y
270  IF X-Y#1 THEN 290
280  PRINT Y"/  "X
290  Y=X
300  X=X+1
310  IF X<M THEN 180
320  PRINT LIN(2)
330  PRINT "THE DIFFERENCES BETWEEN SUCCESSIVE INTEGERS  UP TO";M
340  PRINT "WITH THE GIVEN PRIMES AS THEIR ONLY FACTORS ARE:"
350  PRINT
360  FOR I=1 TO J
370  PRINT D[I];
380  NEXT I
390  PRINT LIN(2)
400  END
```

SAMPLE RUN

```
THIS PROGRAM CAN BE USED TO STUDY SUPERPARTICULAR
RATIOS RELATIVE TO ANY SPECIFIED SET OF PRIMES.

  HOW MANY PRIMES DO YOU WANT TO SPECIFY?3
WHICH ONES ARE THEY (SMALLEST ONE FIRST)?2,3,5
         HOW HIGH DO YOU WANT TO SEARCH ?1000

THE SUPERPARTICULAR RATIOS UP TO 1000
FOR THE SPECIFIED PRIMES ARE:

 1    /    2
 2    /    3
 3    /    4
 4    /    5
 5    /    6
 8    /    9
 9    /    10
 15   /    16
 24   /    25
 80   /    81


THE DIFFERENCES BETWEEN SUCCESSIVE INTEGERS  UP TO 1000
WITH THE GIVEN PRIMES AS THEIR ONLY FACTORS ARE:

 1     1     1     1     1     2     1     1     2     3     1     2
 2     4     1     2     3     2     4     4     5     3     2     4
 6     4     8     3     5     1     9     6     4     8     12    5
 3     7     9     6     10    2     18    12    8     16    9     15
 3     7     6     14    18    12    20    4     36    15    9     16
 5     27    18    30    6     14    12    28    36    24    25    15
 8     27    45    9     21    18    32    10    54    36    60    12


DONE
```

FLOWCHART


START
INPUT PRIMES P(1),...P(K) AND LIMIT M
J=0,Y=1 X=P(1)
N=X
LOOP 2
FOR I=1 TO K
LOOP 1
DOES N/P(I)=INT(N/P(I))?
yes
N=N/P(I)
no
NEXT I
no
DOES N=1?
no
X=X+1
IS X>M?
yes
PRINT DIFFERENCES D(1),..D(J)
STOP
yes
Y=X
no
J=J+1 D(J)=X-Y
DOES X-Y=1?
yes
PRINT Y/X

FLOWCHART NOTES

1. P(1), P(2), ... P(K) are the K specified primes.
2. M is the specified limit up to which the search is carried out.
3. Y and X store successive integers which have the specified primes as their only factors. J counts the numbers of such integers up to M.
4. Loop 1 checks each integer N for prime factors from the specified list.
5. If N has only these primes as factors then it is stored in X.
6. The difference between successive integers which have the specified primes as their only factors is stored in D(J).
7. If the difference D(J) = 1 then Y/X is a superparticular ratio relative to the set of primes P(1), P(2), ... P(K) and is printed out.
8. Loop 2 checks consecutive integers until the limit M is reached.
9. The differences defined in 6 are printed out.

REFERENCES

Bronowski, J. *The Ascent of Man,* Little, Brown and Company, 1973.

Halsey, G.D., and Hewitt, Edwin. "More on the Superparticular Ratios in Music," *American Mathematical Monthly,* 79:1096-1100; December, 1972.

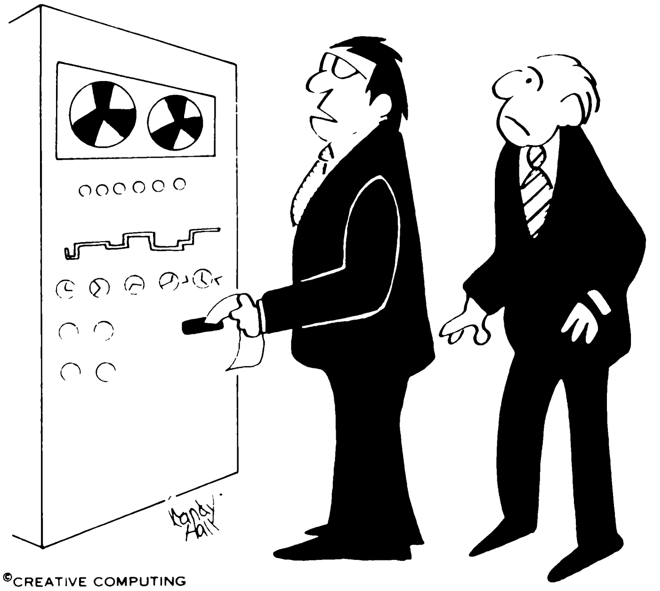
Helm, E.E. "The Vibrating String of the Pythagoreans," *Scientific American,* 217:92-103; December, 1967.

Polya, G. *How To Solve It.* Princeton University Press, 1945.

Silver, A.L. Leigh. "Musimatics or The Nun's Fiddle," *American Mathematical Monthly,* 78:351-57; April, 1971.

Stormer, Carl. "Quelques theoremes sur l'equation de Pell $x^2 - Dy^2 = 1$ et leurs applications." Skrifter Videnskabsselskabet (Christiania) I, Mat.-Naturv. K1., no. 2 (1898), 752-754.

Wickelgren, Wayne A. *How To Solve Problems.* W.H. Freeman and Company, San Francisco, 1974. (Available from *Creative Computing* Book service for $6.75 ppd.)



©CREATIVE COMPUTING

"About our prospects for that merger, it says: 'you have a snowball's chance in . . . .' "

THINKING STRATEGIES WITH THE COMPUTER: WORKING BACKWARD

D.T. Piele and L.E. Wood *

"The so-called 'Treasury of Analysis' is, to put it shortly, a special body of doctrine for the use of those who, after having studied the ordinary Elements, are desirous of acquiring the ability to solve problems."

Pappus, Book VII
Mathematical Collection

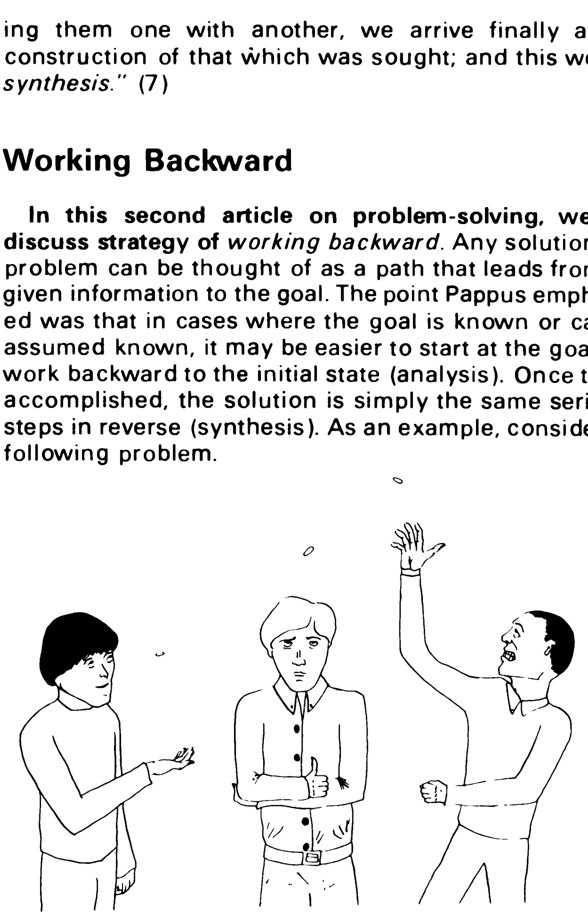
Pappus of Alexandria, who lived at the end of the third century A.D., wrote a comprehensive guidebook and commentary on the geometrical works of the great Greek mathematicians Pythagorus, Euclid, Archimedes, and Apollonius—to name a few. His *Mathematical Collection* consists of eight books describing the important developments of the classical Greek geometers and is punctuated with numerous original propositions, improvements, and historical original propositions, improvements, and historical comments of his own. Book VII is historically very important because it collects together the fundamental discoveries of Greek geometers into a "Treasury of Analysis" which, after Euclid's Elements, became essential reading for serious mathematicians of the day. The "Treasury" is also valuable as an early source for heuristic problem-solving strategies. The strategies of *analysis* and *synthesis* are particularly significant because together they constitute the earliest known description of the problem-solving strategy known today as working backward.

"...for in analysis we assume that which is sought as if it were already done, and we inquire what it is from which this results, and again what is the antecedent cause of the latter, and so on, until, by so retracing out steps, we come upon something already known or belonging to the class of first principles, and such a method we call analysis as being solution backwards.

"But in *synthesis*, reversing the process, we take as already done that which was last arrived at in the analysis and, by arranging in their natural order as consequences what before were antecedents, and successively connecting them one with another, we arrive finally at the construction of that which was sought; and this we call *synthesis*." (7)

Working Backward

In this second article on problem-solving, we will discuss strategy of *working backward*. Any solution to a problem can be thought of as a path that leads from the given information to the goal. The point Pappus emphasized was that in cases where the goal is known or can be assumed known, it may be easier to start at the goal and work backward to the initial state (analysis). Once this is accomplished, the solution is simply the same series of steps in reverse (synthesis). As an example, consider the following problem.



MATCHING COINS

Three men agree to match coins for money. They each flip a coin and the one who fails to match the other two is the loser. The loser must double the amount of money that each opponent has at that time. After three games, each player has lost once, and has $24. How much did each man begin with?

---

*University of Wisconsin-Parkside, Kenosha, Wisconsin 53140

The end result in this problem is known — all three players end up with $24. The initial state can be found by working backward one game at a time. For example, since each player had $24 after the 3rd game, the two winners of this game (who doubled their money) must have had $12 each at the end of the 2nd game. In order to pay each winner $12 and still end up with $24, the loser of this game must have had $48. Thus the distribution of money among the three players after the 2nd game has been determined. In a similar fashion one can continue working backward to reach the initial state.

If we let P_1, P_2 and P_3 represent the players who lost the first, second, and third games respectively, then Figure 1 shows the distribution of money between the three players at each stage constructed by working backward.

States		**Players**	
	P_1	**P_2**	**P_3**
After 3rd game	$24	$24	$24
After 2nd game	$12	$12	$48
After 1st game	$ 6	$42	$24
Initial State	$39	$21	$12

Figure 1. Solution to Matching Coins

Note that in this problem the path from the goal back to the initial state is uniquely determined; thus at each state in the solution, the previous state is forced upon us by the conditions of the problem. By working backward, we were able to arrive at the solution directly without any detours. This property is illustrated in Figure 2.

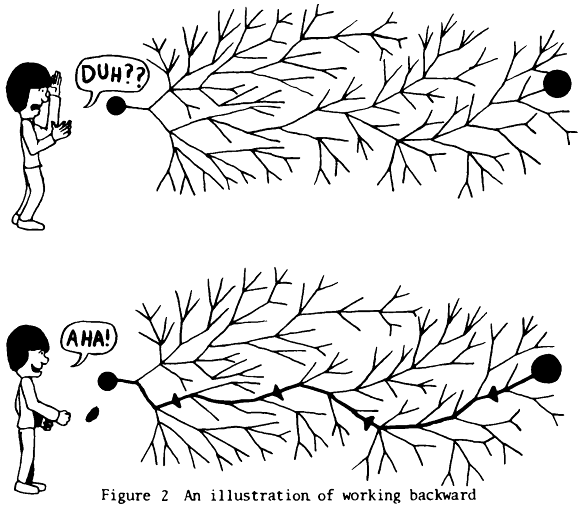


Figure 2 An illustration of working backward

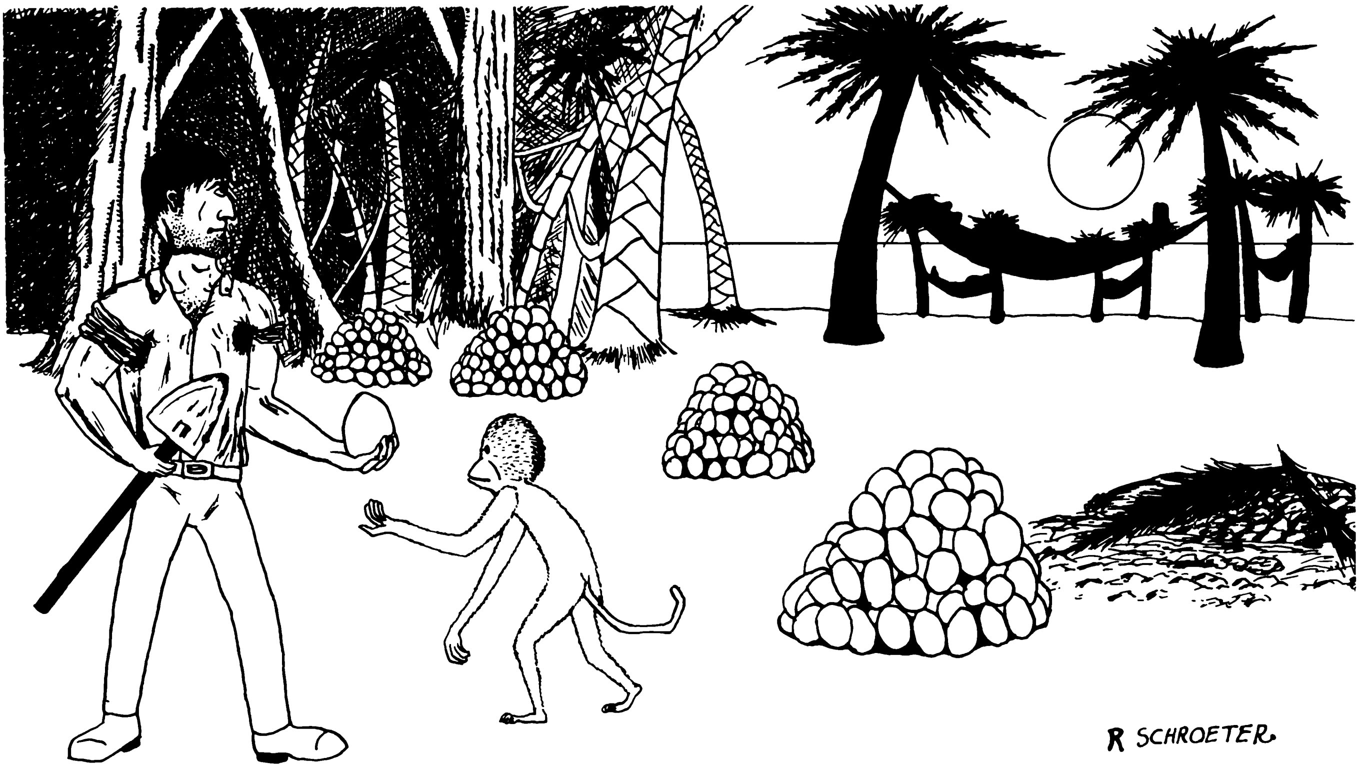
We turn now to a more complex problem where the strategy of working backward is not necessary but where it can be used very effectively in a computer program.

FIVE SAILORS AND A MONKEY

Five sailors and a monkey were on an island. One evening the sailors rounded up all the coconuts they could find and put them in one large pile. Being exhausted from working so hard, they decided to wait and divide them up equally in the morning. During the night, a sailor awoke and separated the nuts into five equal piles, with one left over which he gave to the monkey. He took one pile, hid it, pushed the other four together and went back to his hammock. He was followed in turn by the other four sailors, each of whom did exactly the same thing. Next morning the remaining nuts were divided equally with one remaining nut going to the monkey. What is the least number of coconuts they could have begun with?

Philip W. Brashear (1) derived an elegant algebraic solution to this problem which solves it for any number of sailors. Unfortunately, to conceive such a solution requires a high level of mathematical maturity. But with a computer handy and an understanding of the strategy of working backward, a solution is relatively easy to find.

Consider the general problem where S is the number of sailors on the island and A is the number of coconuts that each sailor received in the final division of the pile. Since one coconut was given to the monkey at every division, the total number of coconuts left in the morning is $S \cdot A + 1$. But this pile came from pushing together $S - 1$ equal piles. Thus, the key condition that must hold is for $(S \cdot A + 1)\ (S - 1)$ to be an integer K, which represents the number of



coconuts that the last sailor stole from a pile of S•K + 1 coconuts. But this pile is the result of pushing together S - 1 equal piles by the previous theif so again (S•K + 1) (S - 1) is an integer as we move back through all S raids on the pile. This idea is explained further in the flowchart and notes which accompany the SAILOR program.

Conclusion

From textbooks, it is easy to get the impression that there is only one way to solve a problem. The trouble is, our memory soon gets overloaded trying to remember which solution goes with which problem and vice-versa. On top of that, what should you do if classical algebraic or analytical techniques become awkward and difficult to solve? Quit? Never!!! Learn a few simple problem solving skills and start cracking some tough coconuts with the computer.

Postscript

The algebraic solution to this problem is given by S(S+1)-(S-1). Thus for S larger than 5, the program given here takes an appreciable amount of time to get an answer. Are there ways to make the program more efficient?

References

1. Brashear, Philip W., "Five Sailors And A Monkey", *The Mathematics Teacher,* October 1967, pp 597-599.

2. Kordemsky, Boris A. *The Moscow Puzzles.* Charles Scribner's Sons. New York 1972.

3. Gardiner, M. *Mathematical Puzzles of Sam Loyd,* Vol. I. Dover Publications, Inc., New York, 1959.

4. Newell, A. and Simon, H.A. *Human Problem Solving.* Prentice-Hall, Inc., Englewood Cliffs, N.J. 1972.

5. Polya, G. *How To Solve It.* Princeton University Press, 1945.

6. Wickelgren, Wayne A. *How To Solve Problems.* W. H. Freeman and Company, San Francisco, 1974.

(7) Heath, Sir Thomas L., *The Thirteen Books of Euclid's Elements,* Dover publications, New York, 1956, p. 138.

Illustrations drawn by Robert Schroeter, a student at UW-Parkside.

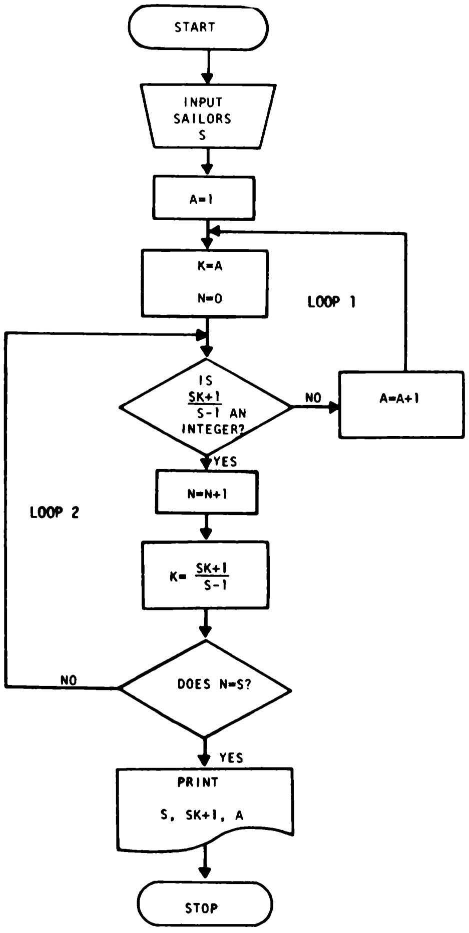
SAILOR PROGRAM

```
LIST
SAILOR

100  PRINT "THIS PROGRAM SOLVES THE SAILORS AND"
110  PRINT "MONKEY PROBLEM BY WORKING BACKWARDS."
120  PRINT
130  PRINT "HOW MANY SAILORS ARE THERE ON THE ISLAND ";
140  INPUT S
150  PRINT
160  A=1
170  K=A
180  N=0
190  IF (S*K+1)/(S-1)=INT((S*K+1)/(S-1)) THEN 220
200  A=A+1
210  GOTO 170
220  N=N+1
230  K=(S*K+1)/(S-1)
240  IF N=S THEN 260
250  GOTO 190
260  PRINT "THE LEAST NUMBER OF COCONUTS THAT"S
270  PRINT "SAILORS CAN BEGIN WITH IS"S*K+1
280  PRINT
290  PRINT "IN THE MORNING, EACH SAILOR GETS"A
300  END
```

Flowchart Notes

1. S is the number of sailors on the island.
2. A = 1 is the initial value for the morning share.
3. K is an integer.
4. N is a counter for loop 2.
5. (S•K + 1) (S - 1) is the number of coconuts stolen by sailor number (S - N) the night before.
6. The value of A is increased by 1 in loop 1 until it reaches a number for the final share that could have come from a pile formed by pushing together S-1 equal shares.
7. Loop 2 checks to see when a number is reached for the final share that can survive being pushed back through S consecutive raids and regroupings and still give integers at each stage.
8. The print-out gives the number of sailors, the least number of coconuts they could have begun with, and the share each sailor received in the morning.

FLOWCHART



SAMPLE RUN

```
THIS PROGRAM SOLVES THE SAILORS AND
MONKEY PROBLEM BY WORKING BACKWARDS.

HOW MANY SAILORS ARE THERE ON THE ISLAND ?5

THE LEAST NUMBER OF COCONUTS THAT 5
SAILORS CAN BEGIN WITH IS 15621

IN THE MORNING, EACH SAILOR GETS 1023

DONE
```

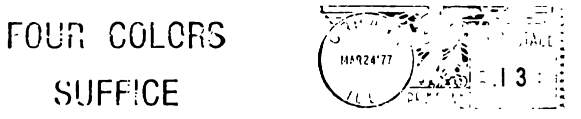
Third in a series, this article shows how to break a problem into simpler subproblems.

THINKING STRATEGIES WITH THE COMPUTER: SUBGOALS

Donald T. Piele and Larry E. Wood*

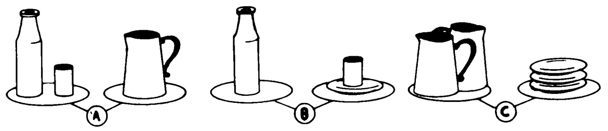
> *Nothing is more important than to see the sources of invention which are, in my opinion, more interesting than the inventions themselves.*
>
> Leibnitz

One of the earliest and most famous problems in the field of topology (a branch of geometry) is the four-color problem. Conjectured by the English mathematician Francis Guthrie in 1850, it states that any map on a plane or a sphere can be colored with at most four colors so that any two countries that share a common boundary are colored differently. All attempts to prove this conjecture had been unsuccessful until last year when it was announced by kenneth Apple and Wolfgang Haken of the University of Illinois that it was indeed true. While listening to Professor Haken outline the proof at a recent colloquium, we were struck by his frequent use of clearly defined problem-solving strategies. Of paramount importance was the strategy of subgoals. After the problem had been represented in the rich language of graph theory, it was broken down into 1,930 subproblems each of which could be routinely solved on a computer. After 1,200 hours of computer time, the announcement was made, as anyone knows who has recently received a letter postmarked from the Mathematics Department at the University of Illinois.

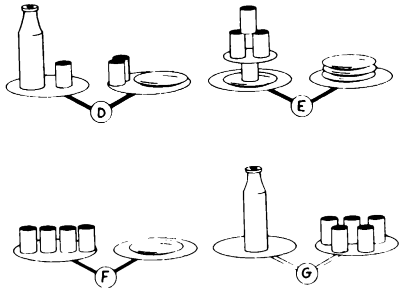


Subgoals

Basically, the method of subgoals consists of breaking a problem into simpler subproblems, solving each part, and regrouping the parts to solve the original problem. We often attack problems this way without thinking of it as a particular strategy since it seems so obvious. However, when we identify this strategy in a variety of problems, we learn how to use it much more effectively. As an example, consider the following balance problem (from Moscow Puzzles). How many glasses will balance a bottle?

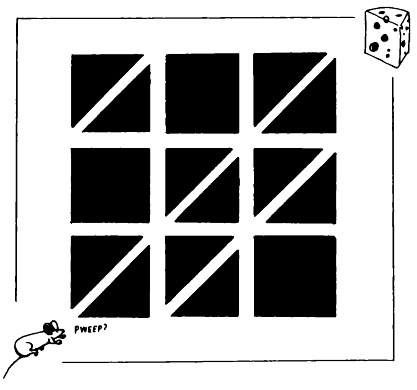


From the information given, it is apparent that the solution will require several step or subgoals. If we spend a few moments actively searching for appropriate subgoals, the solution can be obtained easily. From B, it is obvious that a bottle weighs as much as a glass plus a plate, so to solve the problem it is sufficient to replace the plate by its equivalent weight in glasses. Thus, obtaining a balance between glasses and one plate is a useful subgoal. This relationship is not given explicitly in A, B or C so it is necessary to establish a second subgoal. One possibility is to replace the two pitchers in balance C with glasses and plates. When this second subgoal is achieved, it is possible to reduce the number of plates on both sides until the first subgoal is achieved. The complete solution is:



As a rule, subgoals are easier to attain than the entire goal, and this problem is no exception. Adding a glass to both sides of B yields the balance D. Combining this relationship with A shows that one plate and two glasses balances a pitcher. Hence, the two pitchers in C may be replaced with equivalent plates and glasses as shown in E. This solves the second subproblem which leads to the solution of the first subproblem F and the main problem G.

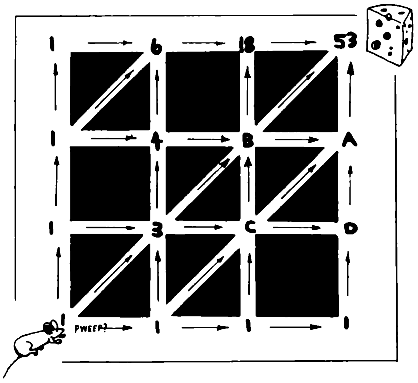
We next turn to an application of the subgoal strategy where recursive relationships can be used. Polya (1957) advises, "If you can't solve the problem posed, try to solve a simpler related problem." Many times the solutions to simpler problems may be combined and expanded in a recursive way to solve the original problem. As an example, consider the following AMAZE problem. A mouse enters a maze in search of a piece of cheese. There are infinitely many paths the mouse could follow but only a finite number will lead the mouse closer to the goal with every step. How many such paths are there?



---

*University of Wisconsin-Parkside, Kenosha, Wisconsin 53140

A poor way to attach this problem is to try to trace all the distinct routes and add them up. A better way is to build from simpler subproblems by placing the cheese at any one of the 14 other intersections. These are certainly related problems since any path that leads to the upper-right-hand corner must pass through a sequence of intersections. Also, the solutions for the simpler problems can be obtained through recursive relationships. There are two AMAZing things about solving the problem this way. The first is that it is really unnecessary to trace all of the paths to count them, and the second is that anyone could solve the problem this way in five minutes or less. For example, the number of paths that lead to intersection A, shown in the next figure, is the sum of the number of paths that lead to B, C and D because the only routes to A are through those intersections. The number of paths to each intersection is found recursively by starting in the lower left hand corner of the figure and moving to the upper right hand corner. As the problem is stated, there are 53 different paths the rat could take to the cheese.

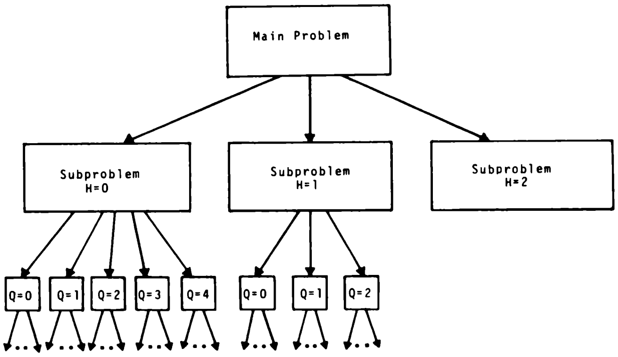


A classic example of the use of recursively defined subgoals appears in the solution to the Tower of Brahma (Hanoi) puzzle (see *Creative Computing,* January-February, 1976 and Wickelgren, 1974).

Change for a Dollar

There are many ways, similar to those above, to apply the subgoal strategy to sole problems with the aid of a computer. A good example appears in the solution to the DOLLAR problem posed for *Creative Computing* by Brian Hess (1976): How many ways can you change a dollar bill?

Begin the solution by searching for ways to divide the problem into a set of smaller subproblems, each of which is easier to solve. One way to do this is illustrated by the tree diagram:



The main problem is broken up into three subproblems: the number of ways of making change for one dollar using,

1. No half dollars (H = 0),
2. One half dollar (H = 1), or.
3. Two half dollars (H = 2).

The last problem is trivial (only one way), while the other two need to be broken down further. This is done by dividing the remaining money into quarters and considering subproblems that specify the number of quarters (Q) used. Continuing on to lower denominations, subgoals are established that specify dimes (D), and nickels (N). As the number of subproblems is expanded, each one becomes easier to solve. In fact, with this problem, subgoals are reached which can be solved in only one way. For example, if H = 1, Q = 1, D = 2, and N = 0, then the pennies (P) must equal five in order to total up to one dollar.

While it is possible to continue the tree diagram in the figure by hand, it is very laborious to do so. However, it is a simple matter to program the computer to keep track of each subproblem with nothing more than nested loops. To demonstrate this, program DOLLAR was written so that each loop breaks the main problem down according to the scheme shown in the figure. Notice that at the quarter stage Q and thereafter, adjustments are made on the limits of the loops depending on how much money is left to change. For example, if H = 1 the only possible subgoals for quarters are Q = 0, 1, or 2 but not 3 or 4. Also, notice that there is no need to test combinations of coins to see whether they add up to $1.00. Simply counting the number of subgoals is sufficient since each one can be solved in only one way, (i.e., once H,Q,D, and N are specified then P must equal 100-50H-25Q-10D-5N).

Recursion Relationships

Another way to attack this problem was suggested by Polya (1957) and uses recursion relationships in a similar way to that shown in the solution to AMAZE. Begin by defining quantities which represent the number of ways to make change for n cents using specified coins.

A_n only pennies
B_n nickles and pennies
B_n nickles and pennies
C_n dimes, nickels, and pennies
D_n quarters, dimes, nickles, and pennies
E_n half dollars, quarters, dimes, nickels, and pennies
The problem is to find E_n for n = 100.

We can distinguish two cases in making change for n cents:

1. No half dollars are used, in which case D_n is the number of ways to change n cents, or
2. One or more half dollars are used. After one half dollar is paid, there remains n - 50 cents to pay which can be done in E_{n-50} ways.

Since these two cases are mutually exclusive, we can infer that

$E_n = D_n + E_{n-50},$

Similarly,

$D_n = C_n + D_{n-25}$

$C_n = B_n + C_{n-10}$

$B_n = A_n + B_{n-5}$

Now we begin with the simplest cases and build up to E_{100} . First of all, it is easy to understand why $E_0 = 1$. From above, when n = 50, $E_{50} = D_{50} + E_0$, and it is possible to make change for 50 cents only *one* more way if half dollars are allowed. Therefore $E_0 = 1$. Likewise, we can argue that $D_0 = C_0 = B_0 = A_0 = 1$. It is also true that $A_n = 1$ for all values of n since there is only one way to make change using only pennies. We are now ready to apply the recursive relationships to solve the original problem. This is the strategy followed in program CHANGE which also has the added advantage that it can count the number of wasy of making change (with coins) for any specified number of cents, n.

Conclusion

Forming subgoals is certainly one of the more common problem-solving strategies. Mathematical induction, recursion, and tree diagrams all contribute to its versatility. When used in conjuction with the computer, this strategy promises applications for solving old problems in new syas and for solving new problems in ways yet to be discovered.

Postscript

The Dollar Problem has been around for some time and can be solved using analytical tcchniques. Kac and Ulam (1968) discuss a solution to this problem using power series. Specifically, if

$P(x) \times 1 + x + x^2 + x^3 + x^4 + \ldots$

$N(x) = 1 + x^5 + x^{10} + x^{15} + x^{20} \ldots$

$D(x) = 1 + x^{10} + x^{20} + x^{30} + x^{40} \ldots$

$Q(x) = 1 + x^{25} + x^{50} + x^{75} + \ldots$

$H(x) = 1 + x^{50}\, x^{100} + x^{150} + \ldots$

then the product series $\pi(x) = P(x)N(x)D(x)Q(x)H(x)$ is the key to finding the number of ways of making change for n cents. For example, the coefficient of the term x^{100} in the product series $\pi(x)$ is the number of ways to make change for a dollar bill. Note, $1 \cdot x^5 \cdot x^{20} \cdot x^{25} \cdot x^{50} = x^{100}$ and this product corresponds to making change for one dollar using a half dollar (x^{50}), a quarter (x^{25}), two dimes (x^{20}), a nickel (x^5) and no pennies (1) and at the same time contributes 1 to the coefficient of x^{100} in the product series $\pi(x)$. However, finding this coefficient by power-series analysis is very tedious and requires a high degree of mathematical sophistication. On the other hand, by making this connection between the coefficients of $\pi(x)$ and changing money, we can turn the table around and use program CHANGE to compute the coefficients for the product series $\pi(x)$ very quickly.

Suppose we expand AMAZE so that a computer program would be necessary for finding a solution quickly. Can you write a program that will handle any specified arrangement of blocks, some which have alleys? If a certain proportion of the blocks has alleys, how should be blocks be arranged so that the number of paths to the goal is maximized?

Russian coins made of copper and nickel come in denominations of 10, 15, 20, 50, and 100 kopecks (100 kopecks = one ruble). Copper-zinc coins come in denominations of 1,2,3, and 5 kopecks. How many ways are there to make change for one ruble?

References

Hess, B. "How Many Ways Can You Change A Dollar?" *Creative Computing,* September-October 1976, p.70.

Kac, M., and Ulam, S. *Mathematics and Logic,* Frederick Praeger, Publishers, New York, 1968, pp. 24-26.

Polya, G. *How to Solve It,* Princeton University Press, Princeton, New Jersey, 1957: 252-253.

"Tower of Brahma," *Creative Computing,* January-February, 1977, p. 25.

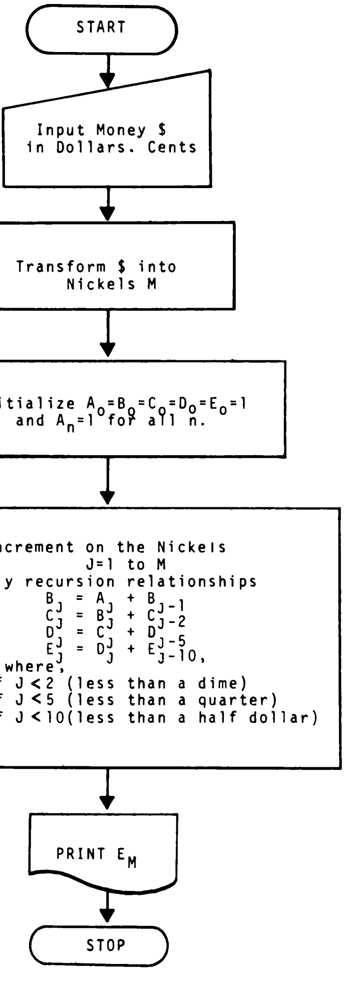
Wickelgreen, AW.A. *How to Solve Problems,* W.H. Freeman and Company, San Francisco, 1974, p. 103.

Illustrations by Rodney Schroeter.

CHANGE PROGRAM

```
10   PRINT 'PROGRAM CHANGE COMPUTES THE NUMBER OF WAYS OF MAKING'
20   PRINT 'CHANGE IN COINS FOR ANY AMOUNT OF MONEY UP TO $5.00.'
30   PRINT
40   DIM A[101],B[101],C[101],D[101],E[101]
50   PRINT 'HOW MUCH DO YOU WANT TO CHANGE?'
60   PRINT 'INPUT $ AS DOLLARS. CENTS ';
70   INPUT C
80   M=INT(20*C)+1
90   A[1]=B[1]=C[1]=D[1]=E[1]=1
100   FOR J=2 TO M
110   A[J]=1
120   B[J]=A[J]+B[J-1]
130   C[J]=B[J]
140   IF J <= 2 THEN 160
150   C[J]=B[J]+C[J-2]
160   D[J]=C[J]
170   IF J <= 5 THEN 190
180   D[J]=C[J]+D[J-5]
190   E[J]=D[J]
200   IF J <= 10 THEN 220
210   E[J]=D[J]+E[J-10]
220   NEXT J
230   PRINT
240   PRINT 'YOU CAN MAKE CHANGE FOR $ ';C
250   PRINT 'IN  ';E[M];'DIFFERENT WAYS.'
260   END
```

CHANGE FLOWCHART



START
Input Money $ in Dollars. Cents
Transform $ into Nickels M
Initialize A0=B0=C0=D0=E0=1 and An=1 for all n.
Increment on the Nickels J=1 to M
Apply recursion relationships
BJ = AJ + BJ-1
CJ = BJ + CJ-2
DJ = CJ + DJ-5
EJ = DJ + EJ-10,
where,
CJ-2 =0 if J<2 (less than a dime)
DJ-5 =0 if J<5 (less than a quarter)
EJ-10=0 if J<10(less than a half dollar)
PRINT EM
STOP

SAMPLE RUN

```
PROGRAM CHANGE COMPUTES THE NUMBER OF WAYS OF MAKING
CHANGE IN COINS FOR ANY AMOUNT OF MONEY UP TO $5.00.

HOW MUCH DO YOU WANT TO CHANGE?
INPUT $ AS DOLLARS. CENTS ?5.00

YOU CAN MAKE CHANGE FOR $  5
IN   59576       DIFFERENT WAYS.

DONE
```

DOLLAR PROGRAM

```
10   REM ***PROGRAM DOLLAR COMPUTES THE NUMBER OF WAYS OF
20   REM ***MAKING CHANGE FOR ONE DOLLAR.
30   C=0
40   FOR H=0 TO 2
50   FOR Q=0 TO 4-2*H
60   FOR D=0 TO 10-5*H-.5*Q
70   FOR N=0 TO 20-10*H-5*Q-2*D
80   C=C+1
90   NEXT N
100   NEXT D
110   NEXT Q
120   NEXT H
130   PRINT "THERE ARE"C;"DIFFERENT WAYS TO CHANGE A DOLLAR BILL."
140   END
```

SAMPLE RUN

```
THERE ARE 292  DIFFERENT WAYS TO CHANGE A DOLLAR BILL.

DONE
```

RAILROAD TALK

Tom Korb

Starting this year, all the nation's railroad cars, from engine to caboose, will have one thing in common. All will have a 12x26 inch color coded information sign attached to their sides.

Chances are, you've probably seen these labels on boxcar sides already, and if you are like most people, thought nothing of them. If however, you were inquisitive, and tried to "break the code", you'd find it next to impossible without a book, as, unlike languages, where certain vowels, words, and phrases are used more often than others, no two cars can have the same identification or configuration. Because of this seemingly disarray of patterns, people tend to think of them as some sort of decoration.

Nothing, of course, could be further from the truth. Every color, every combination of colors, and every position has a meaning. Every one of the thirteen printed modules in a placement is significant to a photometric scanner.

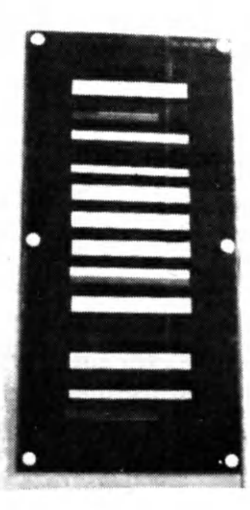
Recorded in the placements are four pieces of vital information; the first being, type of equipment. This can be anything from a locomotive on down, including the piggyback trailers and containers. Next comes the equipment owner, a serial number, and a validation check to verify that all the information is correct.

Because the system is based on the principle of geometric progression, (that is 1-2-4-8-16-32, instead of 1,2,3,4,5,) it is virtually foolproof. In the split second that it takes a railcar to pass by a scanner, a computer translates the colored modules into numbers. The numbers are then multiplied by varying degrees up to the ninth power, added together, and then divided, to get a verification. If any of the modules are ripped off or transposed, the computation will be different than the validation check, and the computer will notify the railroad immediately of a discrepancy.

The amazing part of this system, called AUTOMATIC CAR IDENTIFICATION, (ACI) is the fact that no train will have to be sidetracked to have a man do a car-by-car physical check on them. This will all be done automatically by scanners at checkpoints across the country, and the results put into a central memory bank, where any railroad can get instant information on their cars.

To the rail companies, this means huge savings of both time and money. There will be no more "lost" rolling stock. All equipment will be accounted for, and shipments can be traced on a day by day movement basis. Rush freight can truly be rushed now, as the precise location of cars will be known instantly, just by dialing in the central computer.

To you and me, this will mean faster delivery of goods. Ordered merchandise will get to us faster, and the fresh fruit and vegetables at the store will probably be just a little bit fresher. New jobs will be opening up as the system is expanded. Computer technicians and repair men will be needed as equipment is added and replaced. As always, there will be a need for railroad engineers; but these engineers will be working with slide rules instead of throttles, like Casey Jones did back in the all but forgotten era of steam.



CAR SIGN

This placement is saying to a scanner computer, "I am a caboose belonging to the Chicago and North Western. This is my serial number, and validation check."

Random, systematic and guided trial-and-error strategies are described in this fourth article in the series

Thinking Strategies with the Computer: Trial-and-Error

Donald T. Piele and Larry E. Wood*

> *What is the difference between a method and a device? A method is a device which you use twice."*
> G. Polya

In the course of our formal education, we are taught a great variety of *devices* for solving problems that have already been neatly grouped together at the end of each chapter of a textbook. Typical examples of these devices are formulas, equations, rules, and theorems which are studied for the purpose of attacking certain types of problems. After a careful study of these specific techniques, exams are given to test our ability to recall them. But what do you do when you are faced with the more realistic situation of not being told what device is likely to solve a problem or, worse yet, of having forgotten how to use a technique altogether? Is all hope lost? Of course not, although many students, by the time they reach college, believe that it is.

At the beginning of each semester, we like to ask the students in our freshman and sophomore classes to try to solve a favorite problem of ours by any method they can. It is the Pigs and Chickens problem which appeared in our first article on Thinking Strategies (Piele & Wood, 1977).

PIGS AND CHICKENS

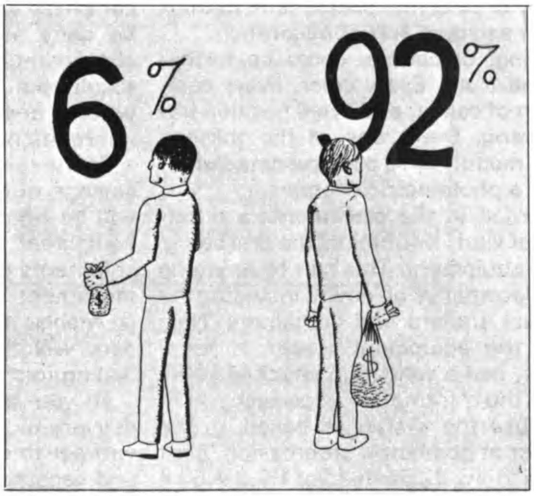
> A boy and his sister visited a farm where they saw a pen filled with pigs and chickens. When they returned home, the boy observed that there were 18 animals in all, and his sister reported that she had counted a total of 50 legs. How many pigs were there in the pen?

Typically the response to this problem is as follows: approximately 40% of the students don't know how to begin because they haven't studied any specific methods for this problem, 25% recognize that the problem could be solved with two equations and two unknowns but have forgotten how to do it, 25% can set up the two equations and get a solution, and 10% quickly try a few numbers and get the answer by trial-and-error in two or three tries. On the other hand, when we give the same problem to elementary-school children who know nothing about two equations and two unknowns and again ask them to solve it using any method they choose, they turn to trial-and-error very naturally and a higher percentage answer it.

What does this all mean? To us it indicates that in the teaching of mathematical and scientific problem-solving in school we overemphasize the memorization of specific devices and techniques for attacking problems, and we underemphasize some very simple and powerful problem solving strategies useful for a variety of problems. In this article on general problem-solving strategies, we would like to turn the tables around and elaborate on the strategy of trial-and-error, which is always available but seldom used to its full potential. It is often frowned upon in school because it is thought to be a lazy approach which requires very little thinking. But with the computer available, trial-and-error takes on a whole new dimension which we will only begin to explore here.

taxes problem

The strategy of trial-and-error can be used in a number of different ways, which we will illustrate with the following problem about income taxes.



---

*University of Wisconsin-Parkside, Kenosha, Wisconsin 53141.

TAXES IN TAXES???

Naturally, many people believe that rich people should pay more taxes than poor people, since the wealthier ones have more money. But sometimes this policy is carried to extremes. In one place I recently heard of, the tax rate was made the same as the number of thousands of dollars a person earns. For example, if a person earns $6,000, then his tax rate is 6% of that. But if a person earns $92,000, then his taxes are a whopping 92% of that.

What income between $1 and $100,000 would leave you the most money after taxes?

One way to solve this problem is *random* trial-and-error. As the term implies, random trial-and-error consists of arbitrarily choosing a series of values (gross salaries in this case) calculating results (net salaries), and then testing to see which one yields the highest value. This method takes little thought and only produces a solution if one happens to pick the correct value or values. In the case of the taxes problem the method would be extremely inefficient because each result must be compared to all previous ones to see if it is larger. It is this type of an approach to problem-solving that has given trial-and-error a bad name.

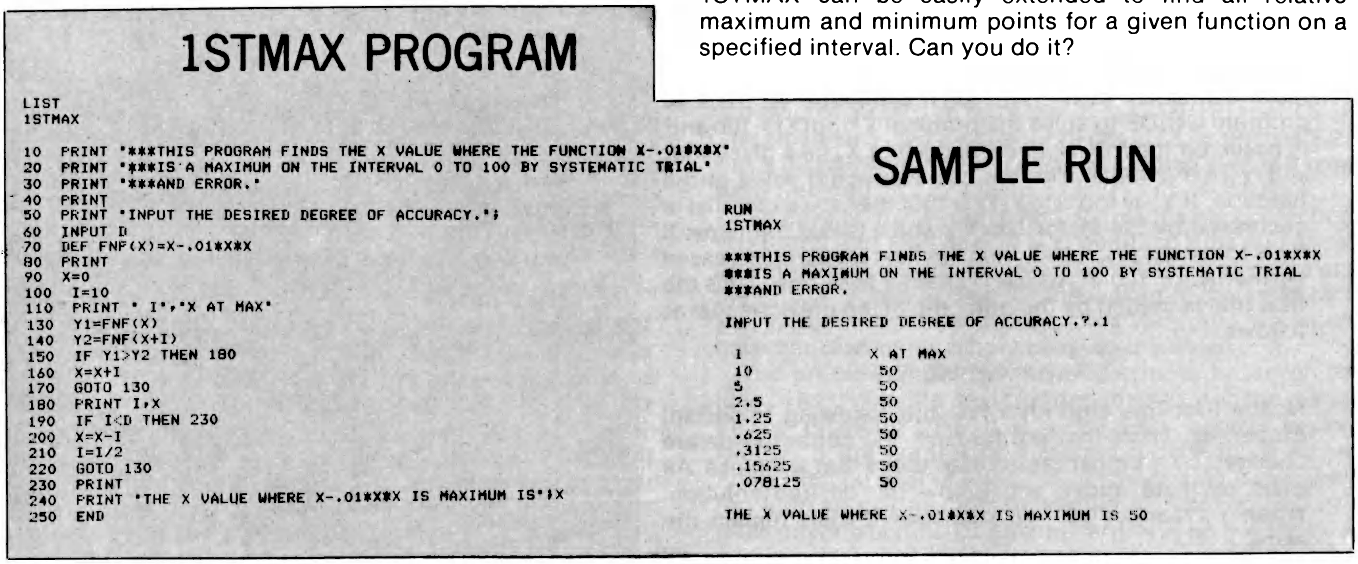
An improvement over random trial-and-error that requires additional thought and substantially improves its utility is *systematic* trial-and-error. Here a rule is devised to make certain all the reasonable alternatives will be systematically considered and evaluated until a problem is solved or shown to be unsolvable. With the tax problem this might consist of beginning with $1,000 and trying successive values in increments of $1,000 until a solution is reached. The result is that the net pay would continue to increase up to a gross income of $50,000 and then begin to decrease. This result implies, of course, that the optimum income is $50,000. While this method may be tedious and time-consuming, it will usually produce a solution, and therefore is a substantial improvement over random trial-and-error.

A further refinement of trial-and-error, and one that makes it much more respectable as a general problem-solving strategy, is *guided* trial-and-error. The key to its success lies in the fact that the results from each trial are used to guide the choice of a value for the next trial that will produce a result closer to the correct solution. This process is continued until the solution is finally attained, and is usually much more efficient than systematic trial-and-error. As an example of guided trial-and-error, let us return to the taxes problem. As a starting point, we might try both values suggested in the problem ($6,000 and $92,000). Because the results show that $92,000 yields a higher income than $6,000, it seems reasonable to choose a value higher than $92,000. As it turns out, however, values higher than $92,000 provide less net income than does $92,000. Therefore, it is logical to choose values less than $92,000 in large increments ($10,000) as long as they continue to result in larger net incomes than values chosen previously. Because the correct answer is $50,000, values closer to $50,000 will produce larger and larger net incomes, and values less than $50,000 will produce decreasing net incomes. Thus, the correct value can be determined quite efficiently. As mentioned earlier, the key to success is to carefully examine the result from each trial to guide the selection of values for the next trial in a way that will guarantee a movement closer to the correct solution.

◆◆◆

systematic trial-and-error

Now we turn to the computer and write a program to find the solution using systematic trial-and-error. If X represents the gross income then .01X is the tax rate and f(X) = (1-.01X)X is the net pay. Program 1STMAX uses a systematic procedure to find the value of X for which the net pay achieves a maximum value on the interval 0 to 100. The system is based on the following principle: Let X=0 be the starting point and increment to the right by I=10, comparing f(X) with f(X+I). If f(X)< f(X+I) then move up one step (X=X+I) and compare f(X) and f(X+I) again. Continue this procedure until f(X)>f(X+I). Now move back one step (X=X-I), reduce the step size by a factor of 2 and continue as before. As soon as the step size falls below the specified level of accuracy D (I<D), print out the value of X that corresponds to the first maximum value of f(X). For functions that have more than one relative maximum, the systematic procedure used in 1STMAX can be easily extended to find all relative maximum and minimum points for a given function on a specified interval. Can you do it?



1STMAX PROGRAM

```
LIST
1STMAX

10  PRINT "***THIS PROGRAM FINDS THE X VALUE WHERE THE FUNCTION X-.01*X*X"
20  PRINT "***IS A MAXIMUM ON THE INTERVAL 0 TO 100 BY SYSTEMATIC TRIAL"
30  PRINT "***AND ERROR."
40  PRINT
50  PRINT "INPUT THE DESIRED DEGREE OF ACCURACY.";
60  INPUT D
70  DEF FNF(X)=X-.01*X*X
80  PRINT
90  X=0
100  I=10
110  PRINT " I","X AT MAX"
130  Y1=FNF(X)
140  Y2=FNF(X+I)
150  IF Y1>Y2 THEN 180
160  X=X+I
170  GOTO 130
180  PRINT I,X
190  IF I<D THEN 230
200  X=X-I
210  I=I/2
220  GOTO 130
230  PRINT
240  PRINT "THE X VALUE WHERE X-.01*X*X IS MAXIMUM IS";X
250  END
```

SAMPLE RUN

```
RUN
1STMAX

***THIS PROGRAM FINDS THE X VALUE WHERE THE FUNCTION X-.01*X*X
***IS A MAXIMUM ON THE INTERVAL 0 TO 100 BY SYSTEMATIC TRIAL
***AND ERROR.

INPUT THE DESIRED DEGREE OF ACCURACY.?.1

 I              X AT MAX
 10              50
 5               50
 2.5             50
 1.25            50
 .625            50
 .3125           50
 .15625          50
 .078125         50

THE X VALUE WHERE X-.01*X*X IS MAXIMUM IS 50
```

x•log(x)=100

Although the taxes problem can be solved analytically using calculus or even more easily with the properties of quadratic functions, it is more likely that the majority of non-textbook problems one encounters will not have a nice closed-form solution. One such example is the following: *Find a value of X such that X • log(X) = 100.* (We assume here that log(X) is the natural logarithm.)

Systematic trial-and-error, which is frequently used to search for solutions with a computer, could be applied to this problem in much the same way it was applied to the taxes problem. But for variety, we will use a different type of systematic trial-and-error.

Clearly, the solution to X• log(X) = 100 lies somewhere between 1 and 100 since 1 • log(1)=0, 100 • log(100)> 400, and X•log (X) increases with increasing values of X. If we let X_l be the guess that is low ($X_l \cdot \log(X\ ) < 100$) and let X_h be the guess that is high ($X_h \cdot \log(X_h) > 100$) then we can use a systematic procedure which generates new trials by dividing the search area in half at each step as follows: Let the new trial X_n be the average of the last two trials $X_n = (X_l + X_h)/2$ and then test X_n to see whether it is high, low, or within the accuracy desired. If X_n is high ($X_n \cdot \log(X_n) > 100$) then replace the last high guess with X_n ($X_h = X_n$) or if X_n is low ($X_n \cdot \log(X_n)\ \ 100$) replace the last low guess with X_n ($X_l = X_n$) and repeat the process of taking averages. Since the distance between X_l and X_h is cut in half with each new trial, X_l and X_h will both approach the desired solution within any pre-set degree of accuracy given a sufficient number of iterations. Program SLICE solves X•log(X) = 100 for X using this method. The sample run following the program lists the upper and lower bounds at each halving of the search area to illustrate the approach. When this printout is suppressed the answer is computed immediately.

SLICE PROGRAM

```
SLICE

10   PRINT "*****THIS PROGRAM SOLVES X*LOG(X)=100 FOR X"
20   PRINT "*****BY USING SYSTEMATIC TRIAL AND ERROR."
25   PRINT
30   PRINT "INPUT A LOWER AND UPPER GUESS AND THE DESIRED ACCURACY";
40   INPUT X1,X2,D
50   N=1
60   Y1=X1*LOG(X1)
70   Y2=X2*LOG(X2)
80   PRINT
90   PRINT " N","X-LOWER","X-UPPER"
100   PRINT N,X1,X2
110   X3=(X1+X2)/2
120   Y3=X3*LOG(X3)
130   IF ABS(Y3-100) <= D THEN 210
140   IF Y1<100 AND Y3>100 THEN 180
150   X1=X3
160   N=N+1
170   GOTO 100
180   X2=X3
190   N=N+1
200   GOTO 100
210   PRINT LIN(1)"THE ANSWER IS "X3
220   END
```

SAMPLE RUN

```
*****THIS PROGRAM SOLVES X*LOG(X)=100 FOR X
*****BY USING SYSTEMATIC TRIAL AND ERROR.

INPUT A LOWER AND UPPER GUESS AND THE DESIRED ACCURACY?1,100,.001

 N              X-LOWER         X-UPPER
 1               1               100
 2               1               50.5
 3               25.75           50.5
 4               25.75           38.125
 5               25.75           31.9375
 6               28.8437         31.9375
 7               28.8437         30.3906
 8               28.8437         29.6172
 9               29.2305         29.6172
 10              29.4238         29.6172
 11              29.5205         29.6172
 12              29.5205         29.5688
 13              29.5205         29.5447
 14              29.5326         29.5447
 15              29.5326         29.5386
 16              29.5356         29.5386
 17              29.5356         29.5371
 18              29.5364         29.5371

 THE ANSWER IS  29.5367
```

guided trial-and-error

The systematic trial-and-error algorithm for solving X• log(X) = 100 given above does not take into account all the information available aftereach trial. For example, this method takes the same amount of time to reach a solution whether the first guess is close to the solution already or not and is independent of the problem being solved. On the other hand, guided trial-and-error uses more of the information available from each trial (such as how close a particular trial is to a solution) to make a more educated next trial. This technique is used in program GUIDE to solve the problem X•log(X) = 100 and is based on the following principle: Let X_n be a given trial and $Y_n = X_n \cdot \log(X_n)$ be the corresponding value of the function. If Y_n is too large ($Y_n > 100$) then the exact trial is decreased by the factor $100/Y_n$ which is less than one. If Y_n is too small ($Y_n < 100$) then the next guess is increased by the factor $100/Y_n$ which is greater than one. Thus the new trial is guided by the outcome of the previous trial as follows:

$$X_{n+1} = X_n{}^* \ 100/Y_n.$$

Notice that this algorithm has the following important properties: Trials that are far from the correct value are changed by a bigger factor than those that are close. All trials oscillate above and below the desired solution. When Y_n reaches 100, all subsequent trials remain the same.

GUIDE PROGRAM

```
GUIDE

10   PRINT "*****THIS PROGRAM SOLVES X*LOG(X)=100 FOR X"
20   PRINT "*****BY USING GUIDED TRIAL AND ERROR."
30   PRINT.
40   PRINT "INPUT AN INITIAL GUESS AND THE DESIRED ACCURACY ";
50   INPUT X,D
60   PRINT
70   PRINT " n"," Xn"
80   N=1
90   Y=X*LOG(X)
100   PRINT N,X
110   IF ABS(Y-100)<D THEN 150
120   X=X*100/Y
130   N=N+1
140   GOTO 90
150   PRINT LIN(1)"THE ANSWER IS"X
160   END
```

SAMPLE RUN

```
RUN
GUIDE

*****THIS PROGRAM SOLVES X*LOG(X)=100 FOR X
*****BY USING GUIDED TRIAL AND ERROR.

INPUT AN INITIAL GUESS AND THE DESIRED ACCURACY ?100,.001

 n              Xn
 1              100
 2              21.7147
 3              32.4887
 4              28.7283
 5              29.7807
 6              29.465
 7              29.5578
 8              29.5303
 9              29.5384
 10             29.536
 11             29.5368

THE ANSWER IS 29.5368

DONE
```

newton's method

For completeness, we conclude our search for solutions to the X•log(X) = 100 problem with Newton's method which is a form of guided trial-and-error. This method is fully explained in almost every calculus book so we will not repeat it here. In comparison with the other algorithms given above, Newton's method is a very efficient approach as shown in the sample run. However, there is a price to pay for this efficiency — a knowledge of calculus and the ability to remember the method — which may not be worth the time and effort in many cases. For example, even though Newton's method converges to the solution over four times faster than the systematic trial and error method of program SLICE, the difference in the response time at the terminal is unnoticeable. Besides, there are some problems where Newton's method will fail if one makes an unlucky first guess, whereas the systematic trial-and-error technique never fails.

NEWTON PROGRAM

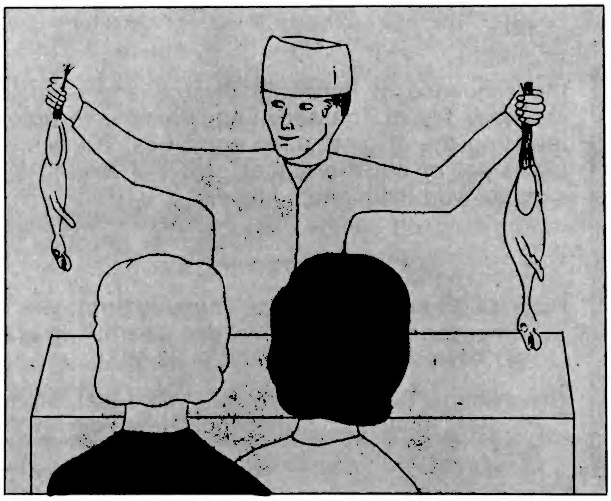
```
NEWTON

10  PRINT "*****THIS PROGRAM SOLVES X*LOG(X)=100 FOR X"
20  PRINT "*****BY USING NEWTONS METHOD FOR GUIDED TRIAL AND ERROR."
30  PRINT
40  PRINT "INPUT AN INITIAL GUESS AND THE DESIRED ACCURACY ";
50  INPUT X,D
60  N=1
70  PRINT
80  PRINT " n"," Xn"
90  PRINT N,X
100  X1=X-(X*LOG(X)-100)/(LOG(X)+1)
110  IF ABS(X-X1)<D THEN 150
120  X=X1
130  N=N+1
140  GOTO 90
150  PRINT LIN(1)"THE SOLUTION IS "X1
160  END
```

SAMPLE RUN

```
RUN
NEWTON

*****THIS PROGRAM SOLVES X*LOG(X)=100 FOR X
*****BY USING NEWTONS METHOD FOR GUIDED TRIAL AND ERROR.

INPUT AN INITIAL GUESS AND THE DESIRED ACCURACY ?100,.001

 n             Xn
 1             100
 2             35.6813
 3             29.6595
 4             29.5367

THE SOLUTION IS  29.5366

DONE
```

turkey puzzle

A completely different application of guided trial-and-error is illustrated in the solution to the *Turkey Puzzle.*



TURKEY PUZZLE

Two people, Jane and Mary, went to the butcher shop to buy turkeys for Thanksgiving dinner. Since Mary had a larger family than Jane, she wanted a larger turkey. The butcher just happend to have two turkeys left, a small one and a large one. "Together these two turkeys weigh twenty pounds," he said. "The little one sells for two cents a pound more than the large one." Jane purchased the little one for 82¢ and Mary paid $2.96 for the big turkey. How much did each gobbler weigh?

Story problems like this one are the bane of most beginning algebra students. They ask "Where do I begin? What method do I use? Did I set it up right?" Frequently, students are primarily concerned about setting up the machinery for a problem so the answer will drop out like an egg into a basket and forget the most fundamental property of any solution: *A solution is an answer that works!* Of course, the Turkey problem can be solved using algebra and the reader may want to try solving it this way. But we will eschew any algebraic devices for attacking this problem to illustrate the power of guided trial and error.

The easiest way to begin the Turkey problem is to try a few numbers. Let's assume, as a first guess, that the small turkey weighs 8 lbs. What implications does this have, given the condition stated in the problem? First, the big turkey must weigh 20 - 8 or 12 lbs. Next, since the big turkey cost $2.96, the price per lb is $2.96/12, about $.25. The small bird cost 2 cents more per lb so its price is $.27. But $.82 was spent for the small bird which means it must have weighed .82/.27, about 3 lbs.

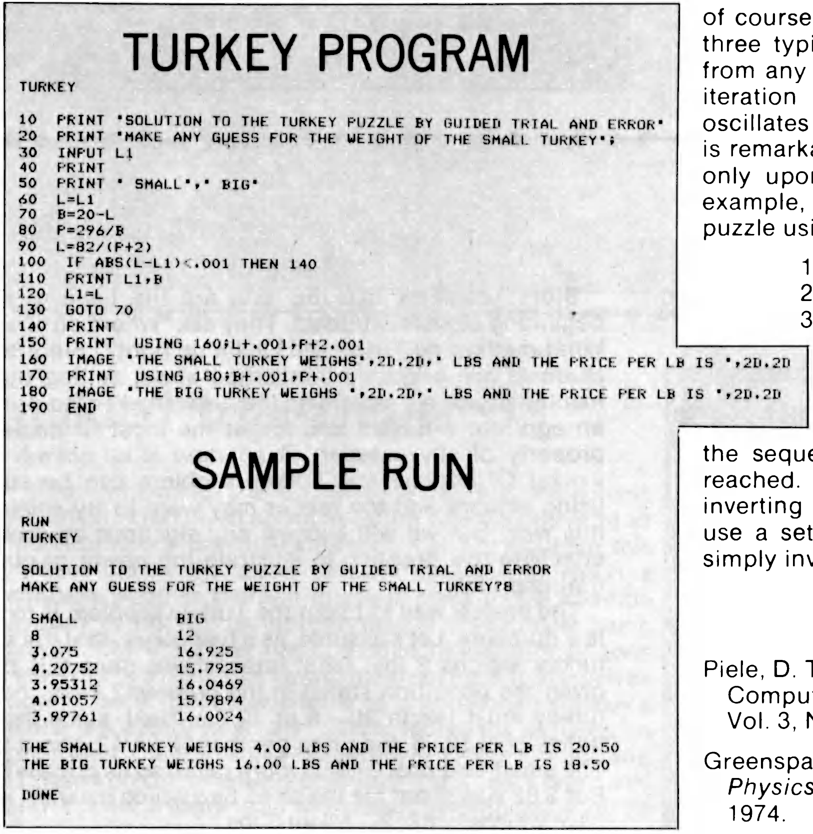
We have come full circle and our results are conflicting. We started out with a small bird weighing 8 lbs and ended up with the same bird weighing 3 lbs. If the first guess and the outcome had agreed for the weight of the small bird, the system would have been consistent and we would have had a solution simply because it worked.

The process of guided trial-and-error is based on the idea of adjusting the next trial depending upon the results of the previous trial. It is a feedback control system similar to the control of a guided missile to its target. From the first trial of 8 lbs the feedback told us we were too high so our next guess should be smaller. In 3 or 4 trials anyone should be able to narrow in on the target of 4 lbs for the small turkey and 16 for the large one.

Would it be possible to set up an automatic trial-and-error or feedback system that is simple enough to be programmed by anyone familiar with the BASIC language and solve the turkey puzzle with only an initial guess? The answer is yes, and program TURKEY is one example of how to do it. The feedback systems of this program follows very closely the first discussion of the Turkey problem given above. If we let L be the weight of the small turkey, B the weight of the large turkey, P the price per lb for the large turkey, then the conditions in the problem can be summarized as follows:

1. B = 20 - L
2. P = 296/B
3. L = 82/(P+2)

Begin with an initial guess of L1 for the weight of the small turkey L and follow through the consequences of this guess in 1, 2, and 3 above. If L1 and L agree, the system is consistent and we have a solution. If not, then use the outcome L in 3 as the next trial in 1 and loop back through the system. This algorithm was used in program TURKEY, and within 7 iterations of the algorithm the system has narrowed in on a solution as shown in the sample run. The program terminates when the difference between successive trials falls below .001.



TURKEY PROGRAM

```
TURKEY

10  PRINT 'SOLUTION TO THE TURKEY PUZZLE BY GUIDED TRIAL AND ERROR'
20  PRINT 'MAKE ANY GUESS FOR THE WEIGHT OF THE SMALL TURKEY';
30  INPUT L1
40  PRINT
50  PRINT ' SMALL',' BIG'
60  L=L1
70  B=20-L
80  P=296/B
90  L=82/(P+2)
100  IF ABS(L-L1)<.001 THEN 140
110  PRINT L1,B
120  L1=L
130  GOTO 70
140  PRINT
150  PRINT  USING 160;L+.001,P+2.001
160  IMAGE 'THE SMALL TURKEY WEIGHS',2D.2D,' LBS AND THE PRICE PER LB IS ',2D.2D
170  PRINT  USING 180;B+.001,P+.001
180  IMAGE 'THE BIG TURKEY WEIGHS ',2D.2D,' LBS AND THE PRICE PER LB IS ',2D.2D
190  END
```

SAMPLE RUN

```
RUN
TURKEY

SOLUTION TO THE TURKEY PUZZLE BY GUIDED TRIAL AND ERROR
MAKE ANY GUESS FOR THE WEIGHT OF THE SMALL TURKEY?8

 SMALL          BIG
 8              12
 3.075          16.925
 4.20752        15.7925
 3.95312        16.0469
 4.01057        15.9894
 3.99761        16.0024

THE SMALL TURKEY WEIGHS 4.00 LBS AND THE PRICE PER LB IS 20.50
THE BIG TURKEY WEIGHS 16.00 LBS AND THE PRICE PER LB IS 18.50

DONE
```

conclusion

The development of digital computers has revolutionized the methods available for solving problems. In many areas of modern science, computer-oriented numerical methods have been developed to solve problems that have been impossible to solve analytically. The diversity of fields being affected includes planetary astrodynamics, wave diffraction, weather prediction, thermodynamics, electrostatics and gravitational potential, molecular interaction, quantum theory, and relativistic collapse (Greenspan, 1974).

The guided trial-and-error methods discussed above are but a small sample of the growing number of computer techniques being used to find solutions or approximate solutions to given mathematical problems. The immense power of the digital computer to perform arithmetical operations with exceptional speed has added a quantum jump in the number of techniques now available to solve problems.

postscript

The iteration technique used in the TURKEY program will converge to the solution, given any initial guess. This of course is the desirable outcome, but it is only one of three typical outcomes. An iteration can diverge away from any solution, getting increasingly larger with each iteration or getting locked into an endless loop that oscillates between two numbers that are not solutions. It is remarkable that the outcome that does occur depends only upon the way the conditions are expressed. For example, if you try writing a program to solve the Turkey puzzle using the following equivalent set of conditions:

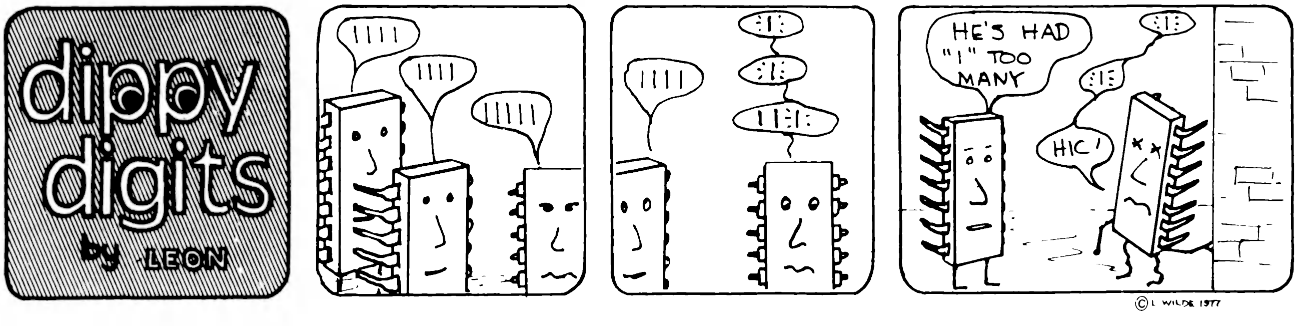
1. L = 20 - B
2. P = 82/L - 2
3. B = 296/P

the sequence of iterations diverge and no solution is reached. Try it! The above equations were obtained by inverting the original three equations. Therefore, if you use a set of equations that give a diverging outcome, simply invert them and try again.

REFERENCES

Piele, D. T., and Wood, L. E. "Thinking Strategies with the Computer: Inference", *Creative Computing Magazine*, Vol. 3, No. 2, Mar-Apr., 1977, p. 96-99.

Greenspan, Donald. *Discrete Numerical Methods in Physics*. Academic Press, Inc., New York, N.Y. 10003, 1974.



© L WILDE 1977

This fifth article in the series shows how indirect reasoning can lead to a contradiction and thus to the truth.

Thinking Strategies with the Computer: Contradiction

Donald T. Piele and Larry E. Wood

Let craft, ambition, spite
Be quenched in Reason's night,
Till weakness turn to might
Till what is dark be light
Till what is wrong be right!
Lewis Carroll

One of the first and possibly most elegant applications of the method of contradiction was given by Euclid in the 3rd century B.C. He proved that among the natural numbers 1, 2, 3, 4, ... there exists an infinite number of primes. (A prime number is one which has no factors other than itself and 1; for example, 2, 3, 5, 7 are prime, while 9 is not.) Euclid's idea is delightfully simple and illustrates the problem-solving strategy of contradiction.

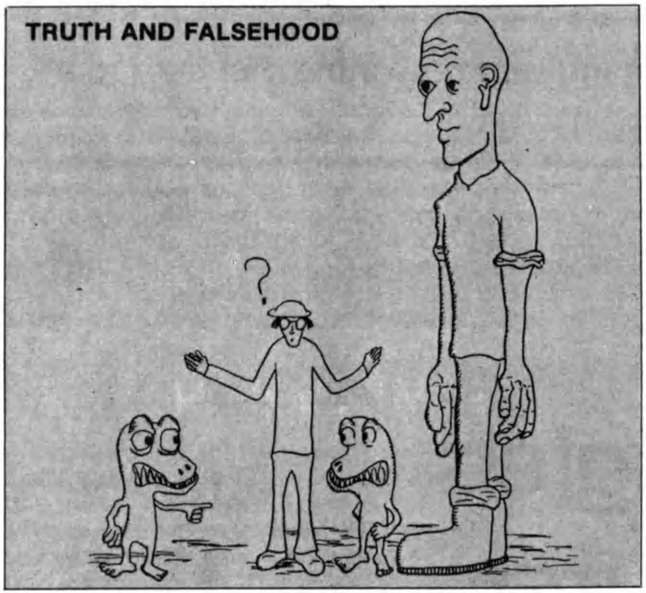
To begin with, there are two possible outcomes:

1. A finite number of natural numbers are prime or
2. An infinite number of natural numbers are prime.

No one has been able to show directly that the second alternative is true. However, by reasoning indirectly, Euclid showed that if you assume the first alternative is true, you can make a sequence of logical inferences that lead to a contradiction. Thus, the first alternative is untenable and the second must be true.

This indirect method called *reductio ad absurdum* is used throughout mathematics and represents one form of the strategy of contradiction. Interested readers may want to study Euclid's clever sequence of logical inferences where he shows that the assumption of only finitely many primes leads to a contradiction (Eves, 1976).

The following problem will be used to illustrate the method of proof by contradiction. The reader is encouraged to try the problem before reading the solution.



TRUTH AND FALSEHOOD

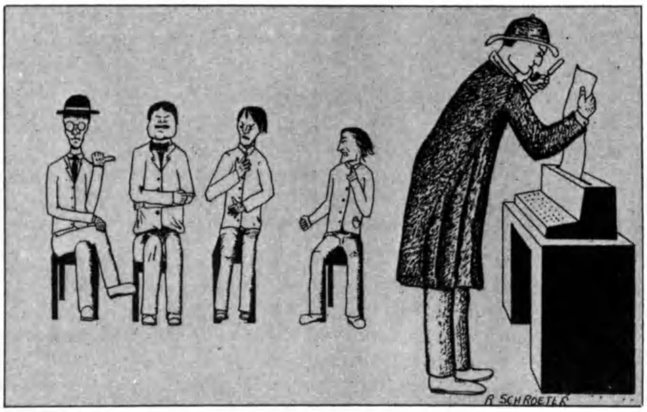
In a faraway land there dwelt two races. The Ananias were inveterate liars, while the Diogenes were unfailingly veracious. Once upon a time, a stranger visited the land, and on meeting a party of three inhabitants inquired to what race they belonged. The first murmured something that the stranger did not catch. The second remarked, "He said he was an Anania." The third said to the second, "You're a liar!"

Now the question is, of what race was this third man?

As in Euclid's problem, there are only two possibilities. The third person is either an Anania (liar) or a Diogene (truar). Let us assume one of the alternatives, he/she is an Anania, and see if we can reach a contradiction. Since the third person said "You're a liar" to the second person and we are assuming the third person is an Anania (liar), it must follow that the second person is really telling the truth (a Diogene). But if the second person is a truar then the statement made about the first person, "He said he was an Anania," is true. But if we examine this statement closely we find that neither a liar or a truar could make it. A liar could not admit to being a liar and a truar would have to say he/she was a Diogene. Thus, we have arrived, by a sequence of logical inferences, to a contradiction based on the original assumption that the third person was a Diogene. Since there are only two alternatives, the third person must be a Diogene and the problem is solved. The reader may want to check that the problem is well-posed and that indeed, if the third person is a truar, the statements made by the first and second persons are non-contradictory.

Now we turn to an application of the method of contradiction where the choice of alternative is more than just two. Whenever the set of alternatives is small enough, it is still feasible to systematically examine each of them and derive a contradiction to all but one. As an example, consider the following problem:

whodunit?



WHODUNIT?

Four men, one of whom is known to have committed a certain crime, said the following when questioned by an inspector from Scotland Yard.

Growley: "Snavely did it."
Snavely: "Gaston did it."
Gus: "I didn't do it."
Gaston: "Snavely lied when he said I did it."

If only one of the four statements is true, whodunit?

Brain teasers like this one often lead the novice, who is likely to attack the problem directly, into an endless loop. The experienced problem-solver recognizes that by first assuming a particular suspect committed the crime, it is an easy matter to check whether this assumption is consistent with the given information or leads to a contradiction. For example, if we assume that Growley did it, we can determine the truth or falsity of the statements given in the problem as follows:

"Snavely did it" is false.
"Gaston did it" is false.
"I didn't do it" is true.
"Snavely lied when he said I did it" is true.

Since two of the suspects are telling the truth and we are given that only one of the first four statements is true, the assumption that Growley did it has lead, through a sequence of logical inferences, to a contradiction. Thus, Growley is innocent. Each suspect, in turn, can be checked out in a similar way. Whenever a problem can be reduced to a bookkeeping chore like this one, it is natural to call in the computer. Even though the work involved in this problem is small and can be easily done by hand, the ideas learned in programming the computer to do the job will be useful when we are faced with a more difficult and time-consuming problem.

We begin the WHODUNIT problem by assigning variables to statements as follows:

P(1)⟷"Growley did it."
P(2)⟷"Snavely did it."
P(3)⟷"Gus did it."
P(4)⟷"Gaston did It."

A statement is designated as true by setting the corresponding variable equal to one, P(I) = 1. If we assume it is false, then P(I) = 0. For example, if Growley is the culprit, then the statement "Growley did it" is true and P(1) = 1, otherwise P(1) = 0. Using this new representation, it is a simple matter to express all the statements (facts of the case) in one equation:

P(2) + P(4) + NOT P(3) + NOT P(4) = 1.

This expression embodies the condition that only one of the statements; "Snavely did it" P(2), "Gaston did it" P(4), "Gus didn't do it" NOT P(3), "Gaston didn't do it" NOT P(4) is true and hence the expression adds to 1. (Recall that in the BASIC language, if A = 1 then NOT A = 0 and vice versa if A = 0 then NOT A = 1.) It is a simple matter for the computer to systematically assume each suspect, in turn, is guilty (for I = 1 to 4, P(I) = 1, and P(J) = 0, $J \neq 1$) and check this assumption for any contradiction with the facts of the case, (P(2) + P(4) + NOT P(3) + NOT (P(4) = 1).

This is the technique employed in program CRIME in lines 80 to 170 to crack the case. The program also checks to see if the problem as posed has a unique solution, no solution, or perhaps many solutions. By changing line 120 (the facts of the case) a new problem can be examined with the same program. The reader is invited to make up new circumstances and test them for solutions. For example, suppose that the statements given by the suspects are: "Growley did it" P(1), "Snavely did it" P(2), "Gaston didn't do it" NOT P(4), "Snavely didn't do it" NOT P(2) and we know that only one of the suspects is lying. Then the facts of the case would be expressed as

$$P(1) + P(2) + \text{NOT } P(4) + \text{NOT } P(2) = 3.$$

Under this assumption, whodunit?

CRIME PROGRAM

```
10   PRINT "THIS PROGRAM SOLVES THE WHODUNIT PROBLEM AS ORGINALLY WRITTEN."
15   PRINT
16   PRINT "FOUR SUSPECTS - GROWLEY, SNAVELY, GUS AND GASTON - ARE QUESTIONED"
17   PRINT "ABOUT A CERTAIN CRIME. THE FOLLOWING STATEMENTS ARE GIVEN AND ONLY"
18   PRINT "ONE IS TRUE."
19   PRINT
20   PRINT "        SNAVELY DID IT."
22   PRINT "        GASTON DID IT."
24   PRINT "        GUS DIDN'T DO IT."
26   PRINT "        GASTON DIDN'T DO IT."
28   PRINT
30   PRINT "THIS IS EXPRESSED IN LINE 120 AS P(2)+P(4)+NOTP(3)+NOTP(4)=1."
32   PRINT
34   PRINT "(YOU CAN MAKE UP YOUR OWN MYSTERY BY CHANGING THE STATEMENTS"
36   PRINT "AND THE CORRESPONDING EXPRESSION IN LINE 120.)"
40   PRINT LIN(2)
50   J=1
60   DIM P[4],A$[72]
70   A$="GROWLEY DID IT.SNAVELY DID IT.GUS DID IT.    GASTON DID IT. "
80   FOR I=1 TO 4
90   MAT P=ZER
100   P[I]=1
110   REM ****************************************
120   IF P[2]+P[4]+ NOT P[3]+ NOT P[4]=1 THEN 150
130   REM ****************************************
140   GOTO 170
150   J=J+1
160   K=I
170   NEXT I
180   GOTO J OF 190,210,230
190   PRINT "SORRY THERE IS NO SOLUTION TO THIS CASE. MORE INFO IS NEEDED."
200   GOTO 250
210   PRINT "THE INESCAPABLE CONCLUSION IS THAT "A$[15*K-14,15*K]
220   GOTO 250
230   PRINT "THERE IS NO UNIQUE SOLUTION TO THIS CASE. MORE THAN ONE SUSPECT"
240   PRINT "IS IMPLICATED BY THE INFORMATION GIVEN."
250   END
```

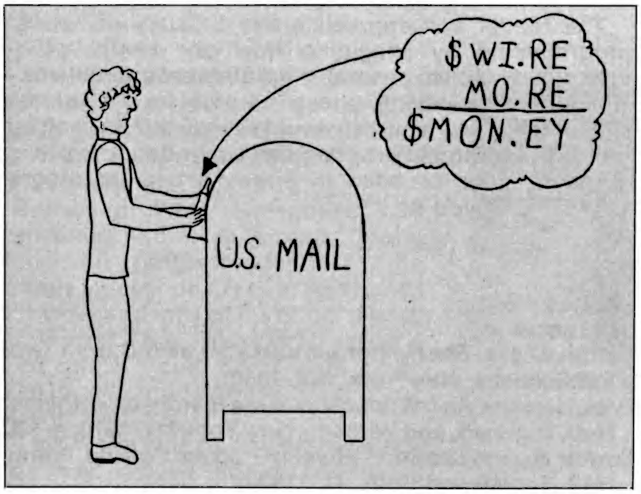
SAMPLE RUN

```
RUN
CRIME

THIS PROGRAM SOLVES THE WHODUNIT PROBLEM AS ORGINALLY WRITTEN.

FOUR SUSPECTS - GROWLEY, SNAVELY, GUS AND GASTON - ARE QUESTIONED
ABOUT A CERTAIN CRIME. THE FOLLOWING STATEMENTS ARE GIVEN AND ONLY
ONE IS TRUE.

        SNAVELY DID IT.
        GASTON DID IT.
        GUS DIDN'T DO IT.
        GASTON DIDN'T DO IT.

THIS IS EXPRESSED IN LINE 120 AS P(2)+P(4)+NOTP(3)+NOTP(4)=1.

(YOU CAN MAKE UP YOUR OWN MYSTERY BY CHANGING THE STATEMENTS
AND THE CORRESPONDING EXPRESSION IN LINE 120.)



THE INESCAPABLE CONCLUSION IS THAT GUS DID IT.

DONE
```

◆

cryptarithmetic

In contrast to the previous examples, where the number of alternatives were relatively small, cryptarithmetic problems usually leave a large number of possible assignments of digits to letters to be examined. For example, consider the following problem.



WIRE MONEY

A college student sent the above message to his father. If each letter represents a unique number, how much should his dad send?

In the jargon of computer science, the *search space* — the set of possible alternatives for considerations as solutions — is 8! or 40,320. It would be impractical in this problem to blindly try each possible assignment one at a time. It would be much more efficient to divide up the search space into large classes, according to a common property shared by members of each class, and then attempt to eliminate entire classes by the method of contradiction. Wickelgren (1974) has labeled this technique *classificatory contradiction.*

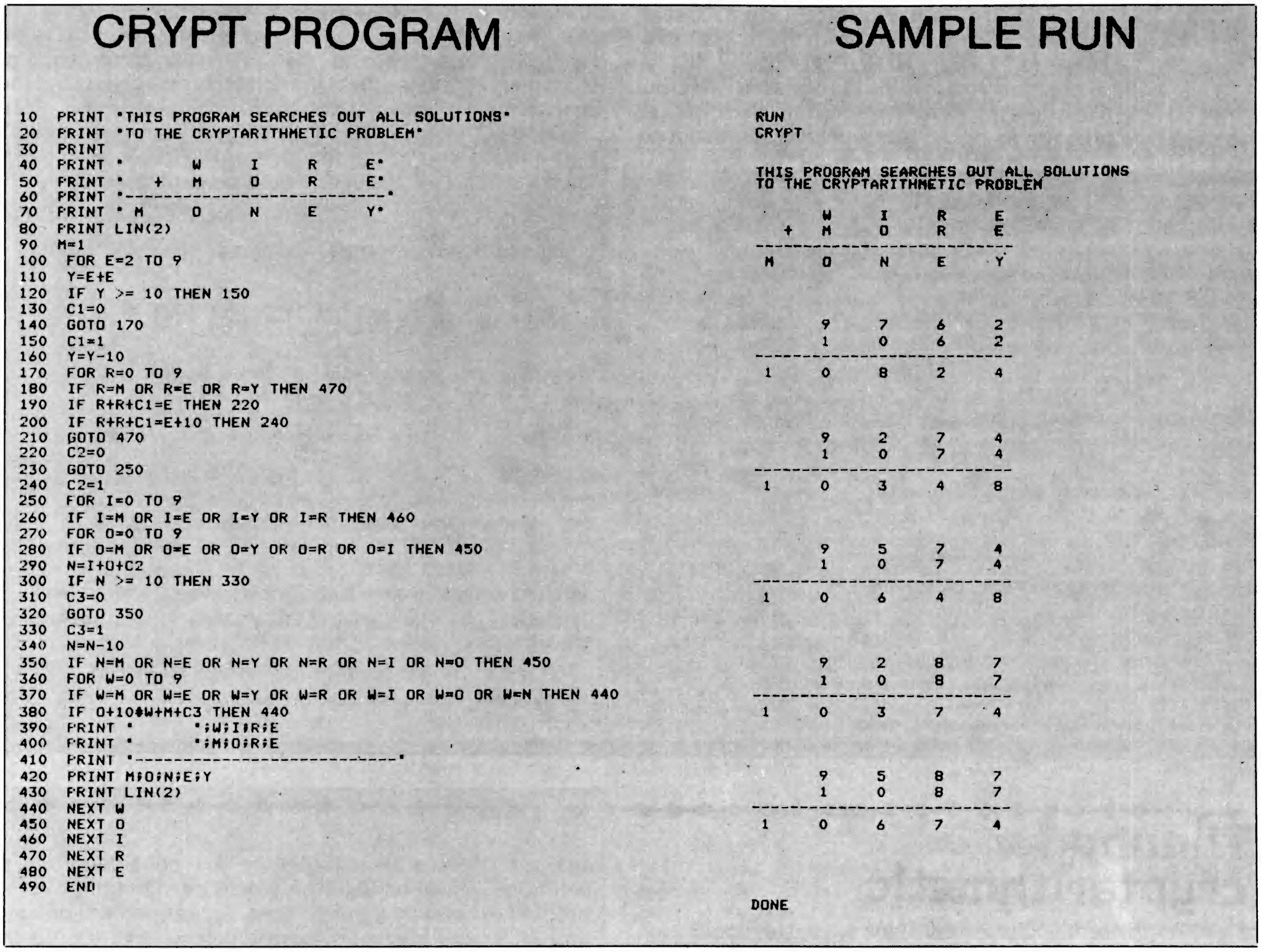
We can illustrate this approach with the problem above by the following argument.

Consider all possible solutions where E = 0. Since E + E = Y, Y also equals 0, contradicting the fact that Y and E must be different digits. Thus, the entire class of solutions where E = 0 is ruled out. For another, less trivial example, consider E = 3. Now Y equals 6 and there is no carry to the next column. Thus in the record column we have R + R = E or perhaps R + R = E + 10 if a carry is involved. But in either case, E would be an even number, since 2R is always even, and this contradicts the assumption that E = 3. Thus the entire class of solutions with E = 3 is ruled out.

By following this same type of classificatory contradiction strategy for each of the digits in order of E, R, I, O, and N, program CRYPT search out all five solutions to the WIRE + MORE = MONEY problem. Can you write your own program to solve the cryptarithmetic problem.

$$\begin{array}{r} \text{DONALD} \\ +\ \underline{\text{GERALD}} \\ \text{ROBERT}\ \ ? \end{array}$$

This problem has been extensively studied by Newell and Simon (1972) using a program called General Problem Solver (GPS) which was written to demonstrate that general problem-solving strategies exist and may be discussed at the very concrete level of computer programming.



CRYPT PROGRAM

```
10   PRINT "THIS PROGRAM SEARCHES OUT ALL SOLUTIONS"
20   PRINT "TO THE CRYPTARITHMETIC PROBLEM"
30   PRINT
40   PRINT "       W     I     R     E"
50   PRINT "   +   M     O     R     E"
60   PRINT "---------------------------"
70   PRINT " M     O     N     E     Y"
80   PRINT LIN(2)
90   M=1
100   FOR E=2 TO 9
110   Y=E+E
120   IF Y >= 10 THEN 150
130   C1=0
140   GOTO 170
150   C1=1
160   Y=Y-10
170   FOR R=0 TO 9
180   IF R=M OR R=E OR R=Y THEN 470
190   IF R+R+C1=E THEN 220
200   IF R+R+C1=E+10 THEN 240
210   GOTO 470
220   C2=0
230   GOTO 250
240   C2=1
250   FOR I=0 TO 9
260   IF I=M OR I=E OR I=Y OR I=R THEN 460
270   FOR O=0 TO 9
280   IF O=M OR O=E OR O=Y OR O=R OR O=I THEN 450
290   N=I+O+C2
300   IF N >= 10 THEN 330
310   C3=0
320   GOTO 350
330   C3=1
340   N=N-10
350   IF N=M OR N=E OR N=Y OR N=R OR N=I OR N=O THEN 450
360   FOR W=0 TO 9
370   IF W=M OR W=E OR W=Y OR W=R OR W=I OR W=O OR W=N THEN 440
380   IF O+10#W+M+C3 THEN 440
390   PRINT "      ";W;I;R;E
400   PRINT "      ";M;O;R;E
410   PRINT "---------------------------"
420   PRINT M;O;N;E;Y
430   PRINT LIN(2)
440   NEXT W
450   NEXT O
460   NEXT I
470   NEXT R
480   NEXT E
490   END
```

SAMPLE RUN

```
RUN
CRYPT


THIS PROGRAM SEARCHES OUT ALL SOLUTIONS
TO THE CRYPTARITHMETIC PROBLEM

         W     I     R     E
     +   M     O     R     E
---------------------------
 M       O     N     E     Y



         9     7     6     2
         1     0     6     2
---------------------------
 1       0     8     2     4



         9     2     7     4
         1     0     7     4
---------------------------
 1       0     3     4     8



         9     5     7     4
         1     0     7     4
---------------------------
 1       0     6     4     8



         9     2     8     7
         1     0     8     7
---------------------------
 1       0     3     7     4



         9     5     8     7
         1     0     8     7
---------------------------
 1       0     6     7     4



DONE
```

conclusion

When it comes to problem solving, one of the most significant advantages that people have over computers is the ease in which humans can make inferences about a problem and quickly reduce the search space. Programming a computer to make similar inferences is at best extremely difficult and at worst, impossible. People can be very unpredictable in ways to survey a problem looking for the easy inferences to attack first, while the computer is easiest to program to attack each problem in a very predetermined and algorithmic way. Polga (1954) contrasts the algorithmic method with the *heuristic* approach in which the nature of the solution is guessed and then proved to be correct. These two approaches have been combined into "heuristic programming" which is a new way of thinking about what a computer program should do. The idea is to find a good set of rules for generating guesses and then prove they are correct. This technique evolved from the work of Newell and Simon (1972) working on the program Logic Theorist (LT) and General Problem Solver (GPS).

The reader can appreciate the difficulty of heuristic programming by imagining how one would write a program to solve general cryptarithmetic problems in which the simplest inferences in a problem are first made similar to the way a human would approach it. In contrast a general algorithmic program can be written to solve any cryptarithmetic problem in a way similar to program CRYPT. Can you do it?

References

Carroll, Lewis. *The Humorous Verse of Lewis Carroll,* Dover Publications, New York, NY, 1960.

Eves, Howard. *An Introduction to the History of Mathematics,* Holt, Rinehart, and Winston, New York, NY, 1976, p. 120.

Newell, A., and Simon, H. *Human Problem Solving,* Prentice Hall, Englewood Cliffs, NJ, 1972.

Wickelgren, Wayne A. *How to Solve Problems,* W.H. Freeman and Company, San Francisco, CA, 1974.

Acknowledgement

We are extremely grateful to Rodney Schroeter for the drawings which he provided throughout the series.

Computer Assisted Instruction (CAI) was one of the earliest and most successful uses of computers in education. In this series of 5 articles we'll show how you can produce and use CAI software on your home or school computer.

CAI: Mathematics Drill and Practice

David H. Ahl

In its most elemental form, CAI presents drill and practice exercises to a student on a subject that he or she has already learned in class or elsewhere. On larger systems this is refined to the point where the computer keeps track of each student and presents proportionately more material of the type with which the student is having difficulty.

For example, in second grade arithmetic a student may receive drill and practice on horizontal addition, vertical addition, horizontal subtraction, and vertical subtraction in equal doses, i.e., 25% of each type of problem. However, over time the student may miss more of the horizontal type problems, particularly subtraction. In this case after several sessions the ratio of problems might be 30% horizontal addition, 15% vertical addition, 40% horizontal subtraction, and 15% vertical addition.

These problem categories are sometimes known as "strands" and a student may progress along each of the strands independently of other strands and independently of his or her overall grade level. Thus, in an extreme case, a third grader could be at sixth grade level in vertical addition and first grade level in fractions.

This is the type of drill and practice that has proved so successful in Chicago, Compton and numerous other places using large-scale computers or dedicated time-sharing mega-minis. However, there's no reason that we can't produce a similar system for micros and minis, or a non-dedicated time sharing system.

Before we produce a relatively elaborate record-keeping system, it's important to understand some of the basics of writing drill and practice for any computer. For example, consider the following problem:

3 + ___ = 17

Where does the student type the answer? With "normal" high-level languages you can request input to the right of the problem or on the next line (a or b). With cursor addressing you could request input at the more desirable location where it actually belongs (c).

3 + **c** ___ = 17 **a**
b

On a problem like this:

$$\begin{array}{r} 32 \\ -\ 17 \\ \hline \end{array}$$

do you require the answer as 15 or do you allow the student to work from right to left, first inputting a 5 and then having the cursor back up for the 1?

Initially, we'll assume the only language available is Basic with no extended capabilities and with no cursor addressing. However, the principles of writing CAI are the same no matter what language you're dealing with.

EXAMPLE 1

```
10 RANDØMIZE
20 N=10
30 W=0
40 A=INT(N*RND(0))
50 B=INT(N*RND(0))
100 PRINT
110 PRINT "     ";A
130 PRINT "  +  ";B
140 R=A+B
200 PRINT "------"
210 INPUT G
220 IF G=R THEN 300
230 W=W+1
240 IF W>1 THEN 270
250 PRINT "WRØNG, TRY AGAIN."
260 GØTØ 100
270 PRINT "YØU MISSED THAT ØNE TWICE."
280 PRINT "THE CØRRECT ANSWER IS ";R
290 GØTØ 310
300 PRINT "CØRRECT !!"
310 PRINT "HERE'S ANØTHER..."
320 GØTØ 30
999 END
```

Numbers in problems will be between 0 and 9.

R = Right answer
G = "Guess" or student input

```
     2
 +   0
------
? 2
CØRRECT !!
HERE'S ANØTHER...

     1
 +   7
------
? 8
CØRRECT !!
HERE'S ANØTHER...

     1
 +   7
------
? 8
CØRRECT !!
HERE'S ANØTHER...

     6
 +   2
------
? 9
WRØNG, TRY AGAIN.

     6
 +   2
------
? 10
YØU MISSED THAT ØNE TWICE.
THE CØRRECT ANSWER IS  8
HERE'S ANØTHER...

     9
 +   1
------
? 10
CØRRECT !!
HERE'S ANØTHER...

     7
 +   8
------
? STØP
PRØGRAM HALTED
```

Two chances to get the correct answer seems about right with young children.

EXAMPLE 2

```
25 P=0
60 P=P+1
70 IF B<=A THEN 100
80 C=A
85 A=B
90 B=C
120 IF P/2=INT(P/2) THEN 160
150 GØTØ 200
160 PRINT "  -  ";B
170 R=A-B
```

Make sure that a smaller number (B) is subtracted from the larger (A).

Alternate between addition and subtraction problems.

```
     6
 +   2
------
? 8
CØRRECT !!
HERE'S ANØTHER...

     8
 -   2
------
? 6
CØRRECT !!
HERE'S ANØTHER...

     7
 +   2
------
? 10
WRØNG, TRY AGAIN.

     7
 +   2
------
? 9
CØRRECT !!
HERE'S ANØTHER...

     9
 -   3
------
? 6
CØRRECT !!
HERE'S ANØTHER...

     7
 +   5
------
? 12
CØRRECT !!
HERE'S ANØTHER...

     7
 -   4
------
? 3
CØRRECT !!
```

Example 1 generates and presents vertical addition problems. It doesn't keep score, it doesn't use cursor addressing, it doesn't have timing, it doesn't even keep columns of numbers lined up, but it's a starting point. And, incidentally *it is useful.* Children are incredibly adaptable and it's frequently easier to get a child to accept a less-than-beautiful format on the computer than to go through the programming gyrations to get everything "just right." The *really important* reasons that CAI is so successful is that it is personal, it is self-paced, it is not critical (in an ego deflating or destructive way), and it is infinitely patient. All these factors are present in Example 1 even though it lacks the niceties of more sophisticated programs.

Notice the following features:

- *Problem difficulty.* This is set in Statement 20. Currently the number range is between 0 and 9. N determines the upper range of numbers used in problems.
- *Number of trials allowed.* Statement 230 counts the number of times a problem is gotten wrong. Statement 240 allows two trials; if you wish to allow 3 trials before giving the correct answer, then Statement 240 should be IF W 2 THEN 270.

By adding 10 statements to Example 1 (see Example 2) we can present addition and subtraction problems alternately. Statement 60 is a problem counter; Statement 120 branches to subtraction problems on even numbers. Statements 70 through 90 simply make sure that a smaller number is being subtracted from a larger one (not necessary, of course, if the student understands the concept of negative numbers).

Example 3, for multiplication problems adds two additional features not in Examples 1 or 2:

- *Personal feedback.* The child's name, input in Statement 10, is used liberally in comments throughout the exercise (Statements 114,145,220).
- *Scoring.* Variable R counts the number of problems right on the first or second trial and Statements 210-220 compute the total score.

EXAMPLE 3

```
2 RANDØMIZE
5 PRINT "MULTIPLICATIØN PRACTICE."
10 PRINT "YØUR NAME";
12 INPUT A$
14 PRINT\PRINT "HI ";A$;".  TØ STØP, TYPE 999 FØR YØUR ANSWER."
15 R=0\P=-1
20 P=P+1
25 Q=0
30 A=INT(12*RND(0)+1)
40 B=INT(10*RND(0)+1)
50 PRINT
60 PRINT "     ";A
70 PRINT "  X  ";B
80 PRINT "  ------"
90 PRINT "  ";
 100 INPUT G
110 IF G=A*B THEN 140
111  IF G=999 THEN 200
112  IF Q<1 THEN 122
114 PRINT "YØU MISSED THAT ØNE TWICE,";A$
115 PRINT "THE CØRRECT ANSWER IS ";A*B
117 PRINT
120 GØTØ 20
122 Q=Q+1
125 PRINT "NØ.  TRY AGAIN."
130 GØTØ 50
140 R=R+1
145 PRINT "RIGHT, ";A$
150 GØTØ 20
200 PRINT
210 PRINT "YØU GØT";R;"RIGHT ØUT ØF";P;"PRØBLEMS."
215 S=INT(100*R/P)
220 PRINT "SCØRE IS";S;"PERCENT, ";A$
230 END
```

Personalization is nice if your Basic has string variables.

The first number will vary between 1 and 12; the second between 1 and 10.

Count correct answers

Present score.

```
MULTIPLICATIØN PRACTICE.
YØUR NAME? DEREK

HI DEREK.  TØ STØP, TYPE 999 FØR YØUR ANSWER.

     9
  X  7
  ------
  ? 54
NØ.  TRY AGAIN.

     9
  X  7
  ------
  ? 64
YØU MISSED THAT ØNE TWICE,DEREK
THE CØRRECT ANSWER IS  63


     2
  X  3
  ------
  ? 6
RIGHT, DEREK

     11
  X  7
  ------
  ? 77
RIGHT, DEREK

     5
  X  4
  ------
  ? 20
RIGHT, DEREK

     10
  X  8
  ------
  ? 80
RIGHT, DEREK

     4
  X  6
  ------
  ? 24
RIGHT, DEREK

     2
  X  3
  ------
  ? 6
RIGHT, DEREK

     1
  X  1
  ------
  ? 999

YØU GØT 6 RIGHT ØUT ØF 7 PRØBLEMS.
SCØRE IS 85 PERCENT, DEREK
```

This program does not right justify digits, however, most children seem to be able to adjust to this format.

Scoring, like grades may not be desirable. It all depends upon your point of view.

Let's now take a bigger jump to Example 4 which presents 9 different types of horizontal and vertical addition and subtraction problems Starting with the same basics, we've added some additional features:

- *Digit alignment* in vertical problems. Statements 114, 115, and 401 determine how many spaces to tab over (Statements 210, 220, etc.) so that the digits are right justified.
- *Different reinforcement messages.* Problem counter Y coupled with Statements 750-795 alternates between 4 reinforcement messages. More could be used, of course.

Notice that scoring is not in this program. Scoring is most valuable when it is an internal variable used to alter the ratio of different types of problems in response to what the child is getting right or wrong. However many children feel threatened by scores (like grades) so it may not be desirable to print it out.

Next issue we'll look at how the scores on different types of problems can be used to vary the ratio of problem types presented and we'll also look at keeping records from one session to the next. ■

EXAMPLE 4

```
10 RANDØMIZE
20 N=15
30 PRINT "HI.  WHAT'S YØUR NAME";
40 INPUT A$
50 PRINT
60 PRINT "ØK, "A$", WE'RE GØING TØ DØ SØME ARITHMETIC PRØBLEMS."
80 E=0
85 Y=0
90 FØR P=1 TØ 18
100 A=INT(N*RND(0)+1)
105 B=INT(N*RND(0)+1)
110 IF A>B THEN 114
111  D=B
112  B=A
113  A=D
114 S=3-INT(LØG(A)/2.302585+1)
115 T=2-INT(LØG(B)/2.302585+1)
120 Q=P
130 IF P<10 THEN 150
140 Q=P-9
150 ØN Q GØTØ 200,250,300,350,400,450,500,550,600
200 R=A+B
210 PRINT TAB(S);A
220 PRINT TAB(T)"+"B
225 PRINT "------"
230 INPUT G
235 GØSUB 700
240 IF E>0 THEN 210
245 GØTØ 680
250 R=A-B
260 PRINT A" - "B" = ";
270 INPUT G
280 GØSUB 700
 290 IF E>0 THEN 260
295 GØTØ 680
```

Numbers in problems vary between 1 and N.

Determines the width of a number.

Alternates between the 9 problem types.

Vertical addition
11
+ 2

Horizontal subtraction 12 - 4 = ___

```
300 R=A-B
310 PRINT TAB(T)" "B
320 PRINT "+"
 325 PRINT "------"
330 PRINT TAB(S);A, ,
332 INPUT G
335 GØSUB 700
 340 IF E>0 THEN 310
345 GØTØ 680
350 R=A-B
360 PRINT A" -              = ";B,,
370 INPUT G
380 GØSUB 700
 390 IF E>0 THEN 360
395 GØTØ 680
400 C=INT(N*RND(0)+1)
401 U=3-INT(LØG(C)/2.302585+1)
402 R=A+B+C
410 PRINT TAB(S);A
415 PRINT TAB(U);C
420 PRINT TAB(T);"+";B
425 PRINT "------"
430 INPUT G
435 GØSUB 700
440 IF E>0 THEN 410
445 GØTØ 680
450 R=A+B
460 PRINT A" + "B" = ";
470 INPUT G
480 GØSUB 700
490 IF E>0 THEN 460
495 GØTØ 680
500 R=A-B
510 PRINT TAB(S);A
520 PRINT TAB(T)"-"B
525 PRINT "------"
530 INPUT G
535 GØSUB 700
540 IF E>0 THEN 510
545 GØTØ 680
550 R=A-B
560 PRINT B" +          = "A,,
570 INPUT G
580 GØSUB 700
590 IF E>0 THEN 560
595 GØTØ 680
600 R=A-B
610 PRINT TAB(S);A
615 PRINT "-"
 620 PRINT "------"
625 PRINT TAB(T)" "B,,
630 INPUT G
635 GØSUB 700
640 IF E>0 THEN 610
645 GØTØ 680
680 NEXT P
690 GØTØ 900
700 IF G=R THEN 750
705 E=E+1
710 IF E>2 THEN 800
720 PRINT "WRØNG.  TRY AGAIN."
725 PRINT
730 RETURN
750 Y=Y+1
752 E=0
755 ØN Y GØTØ 760,770,780,790
760 PRINT "VERY GØØD "A$
765 GØTØ 725
770 PRINT "SUPER !"
775 GØTØ 725
780 PRINT "THAT'S RIGHT "A$
785 GØTØ 725
790 PRINT "CØRRECT !"
792 Y=0
795 GØTØ 725
800 PRINT "YØU MISSED THAT ØNE 3 TIMES "A$"."
805 PRINT "THE CØRRECT ANSWER IS "R" ."
810 PRINT "HERE'S ANØTHER PRØBLEM."
815 E=0
820 GØTØ 725
900 PRINT
910 PRINT "THAT WAS LØTS ØF FUN "A$"."
920 PRINT "DØ YØU WANT ANY MØRE PRØBLEMS TØDAY (YES ØR NØ)";
930 INPUT B$
940 IF B$="YES" THEN 85
950 IF B$="NØ" THEN 960
953 PRINT "PLEASE ANSWER 'YES' ØR 'NØ'."
955 GØTØ 920
960 PRINT
970 PRINT "ØK.  GØØDBYE FØR NØW "A$". PLEASE TYPE 'BYE' AND"
980 PRINT "HANG UP THE PHØNE.  THANKS."
999 END
```

Vertical addition (type 2) 2 + ___ 7

Horizontal subtraction (type 2) 8 - ___ = 5

Vertical addition 10 3 10 ___

Horizontal addition 15 + 1 = ___

Vertical subtraction 15 -15 ___

Horizontal addition (type 2) 2 + ___ = 11

Vertical subtraction (type 2) 10 - ___ 10

Subroutine to check answer (G) against correct one (R). E counts the number of incorrect trials.

Alternates between 4 reinforcement messages.

```
HI.  WHAT'S YØUR NAME? DETTA

ØK, DETTA, WE'RE GØING TØ DØ SØME ARITHMETIC
  11                                 PRØBLEMS.
 + 2
------
? 13
VERY GØØD DETTA

 12  -  4  = ? 8
SUPER !

  2
+
------
  7                          ? 5
THAT'S RIGHT DETTA

 8  -        =  5            ? 3
CØRRECT !

  10
   3
+ 10
------
? 23
VERY GØØD DETTA

 15  +  1  = ? 16
SUPER !

  15
- 15
------
? 1
WRØNG.  TRY AGAIN.

  15
- 15
------
? 10
WRØNG.  TRY AGAIN.

  15
- 15
------
? 20
YØU MISSED THAT ØNE 3 TIMES DETTA.
THE CØRRECT ANSWER IS  0 .
HERE'S ANØTHER PRØBLEM.

 2  +        =  11           ? 9
THAT'S RIGHT DETTA

  10
-
------
  10                         ? 1
WRØNG.  TRY AGAIN.

  10
-
------
  10                         ? 0
CØRRECT !

  13
 + 6
------
? 19
VERY GØØD DETTA

 5  -  1  = ? 4
SUPER !

  10
+
------
  13                         ? 3
THAT'S RIGHT DETTA

 13  -       =  8            ? 5
CØRRECT !

  12
   5
 + 2
------
? 19
VERY GØØD DETTA
```

Without cursor addressing, the answer required here must be input here.

Answers to problems like this must be input as 23, not 3 then 2 which may be what the student is used to.

The second in this five-part series on Computer Assisted Instruction shows how the score can be used to tailor the type and difficulty of problems to the individual student.

CAI: Structuring the Lesson to the Student

David H. Ahl

In Part 1 of this series we looked at how various types of arithmetic problems can be generated. In this section we'll look at the use of scoring to vary the ratio or type and difficulty of problems.

Let's look at the performance of 3 students on 4 types of problems. We'll assume that each student has completed 100 problems, 25 of each type. Here are the percentage missed (incorrect) of each type of problem.

Problem Type	*Student* 1	2	3
Vertical addition	0%	8%	24%
Horizontal addition	4	12	32
Vertical subtraction	0	12	36
Horizontal subtraction	8	20	48

It is obvious that Student 1 is doing quite well—3 problems wrong out of 100. It is equally obvious that Student 3 is having trouble, with 35 out of 100 problems incorrect. Student 2 is somehewere in between.

Now our situation is that we'd like to alter the ratio of problems presented to each student in order to give them additional practice on the types with which they're having trouble. A straightforward way of doing this is to come up with a ratio based on problems missed previously (or in a pre-test). Let's try this for Student 2:

Problem Type	*% Wrong*	*Ratio*	*New Distribution of Problems*
Vertical addition	8%	2	16.7%
Horizontal addition	12	3	25.0
Vertical subtraction	12	3	25.0
Horizontal subtraction	20	4	33.3
		12	100.0%

Good approach? Well, maybe. Except that Student 1 would get 67% horizontal subtraction and 33% horizontal addition and no other types. How then does the student advance to a higher grade-level in vertical subtraction? Clearly, we've overlooked something. But before discussing this, let's consider another factor.

If a student misses 5 out of 5 fraction problems on Monday and then misses 0 out of 5 on Tuesday, what does that mean? Does this mean:

(1) The overall score is 50% and the student needs more practice.

(2) The student learned the concept after a dismal performance Monday and now needs no further practice.

```
HI.  TO STOP, TYPE 9999 FOR YOUR ANSWER.
WHAT IS YOUR GRADE LEVEL? 2.5

   150
 +  33
-------
? 183
CORRECT !!

HERE'S ANOTHER...

    38
 +  54
-------
? 82
WRONG, TRY AGAIN.

    38
 +  54
-------
? 92
CORRECT !!

HERE'S ANOTHER...

    30
 +  68
-------
? 98
CORRECT !!

HERE'S ANOTHER...

   100
 +  94
-------
? 194
CORRECT !!

HERE'S ANOTHER...

   135
 +  55
-------
? 190
CORRECT !!

HERE'S ANOTHER...

    13
 +  52
-------
? 65
CORRECT !!

HERE'S ANOTHER...

     6
 + 119
-------
? 9999
OKAY.  SO LONG.  HOPE YOU ENJOYED IT.
YOUR GRADE LEVEL IS NOW  2.55943
```

Student inputs his grade level from the last session.

These 3 problems helped the student advance his grade level slightly.

(3) The student lucked out on Tuesday and got a batch of trivial problems.

What would be the explanation if the scoring was reversed; that is 5 correct Monday and 5 missed Tuesday? The point is, and this ties in with the previous scoring situation, that we need a moving-average type of scoring system which meets the following criteria:

(1) It weighs the most recent performance most heavily, but does not ignore previous performance.

(2) It allows a student to advance to more difficult problems than their current mastery level.

(3) It continues to give some minimal practice on problem types the student has already mastered.

(4) It is simple to understand for both student and teacher (or parent or administrator).

If we proceed along traditional lines, we'll have to determine what type of problems a student should be receiving practice in, his score on each in the last session, and his score in the sessions before that—a tricky bit of record-keeping. But what if we could come up with a single measure for each type of problem, say "estimated grade level," which incorporated all of the above measures?

Consider the following "scores" for Derek Carlson:

	Grade Level						
	1	2		3	4	5	6
Vertical addition			X				
Horizontal addition		X					
Vertical subtraction				X			
Horizontal subtraction		X					

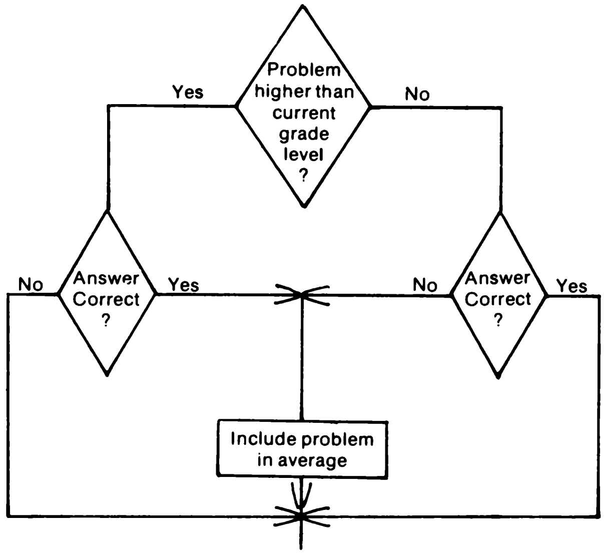
If Derek is halfway through Grade 2, he is behind in 2 problems types, ahead in 1 and on-target with 1. The nice thing for us is we have all the information we need to give him more problems at the "right" level. Naturally this assumes we know what problems of what complexity are being done at every grade level.

So how do we score? Simply by letting the most recent problem done count 10% of the overall score (of that problem type) if the problem was correct and over his current grade level or if it was incorrect and under his current level. The problem is ignored if he got it wrong and it was over his current level, or got it right and it was under his grade level. In other words:

		Answer	
		Right	*Wrong*
Problem	*Higher than grade level*	Raise student grade level	Ignore
	Lower than grade level	Ignore	Lower student grade level

At first glance this may look complex and even somewhat goofy, however what it really means is that is that a student is "rewarded" for doing a problem beyond his grade level but he is not penalized if he can't do it. On the other hand he is penalized if he can't do a problem lower than his grade level, but not rewarded for doing one lower.

A flowchart of this process is shown below.


Problem higher than current grade level ?
Yes
No
Answer Correct ?
No
Yes
Answer Correct ?
No
Yes
Include problem in average

```
10 RANDØMIZE
15 PRINT "HI.  TØ STØP, TYPE 9999 FØR YØUR ANSWER."
20 PRINT "WHAT IS YØUR GRADE LEVEL";
30 INPUT G1
40 G2=G1-.5+RND(0)
50 R=INT(2*1.73*G2^4)
60 A=INT(100*G2*RND(0))
70 IF A>R THEN 60
80 B=R-A
100 PRINT
110 PRINT "    ";A
120 PRINT "  + ";B
140 R=A+B
200 PRINT "-------"
210 INPUT G
220 IF G=R THEN 310
225 IF G=9999 THEN 500
230 W=W+1
240 IF W>1 THEN 270
250 PRINT "WRØNG, TRY AGAIN."
260 GØTØ 100
270 PRINT "YØU MISSED THAT ØNE TWICE."
280 PRINT "THE CØRRECT ANSWER IS ";R
285 W=0
290 IF G2>G1 THEN 400
300 GØTØ 350
310 W=0
320 PRINT "CØRRECT !!"
330 IF G2<G1 THEN 400
350 G1=.9*G1 + .1*SQR(SQR(R/2/1.73))
400 PRINT \ PRINT "HERE'S ANØTHER..."
410 GØTØ 40
500 PRINT "ØKAY.  SØ LØNG.  HØPE YØU ENJØYED IT."
510 PRINT "YØUR GRADE LEVEL IS NØW ";G1
999 END
```

G2, the grade level of the problem to be done is ± ½ a grade of the student's level, G1.

Statements 50 and 60 calculate the addends of the problem.

Statement 350 gives us a moving grade level average.

We said that if a problem is to be counted in the student's overall average, it counts 10% of the total. If his current grade level (on a particular problem type) is L and the level of the problem to be averaged in is P, then the averaging formula is simply:

$$L = .9L + .1P$$

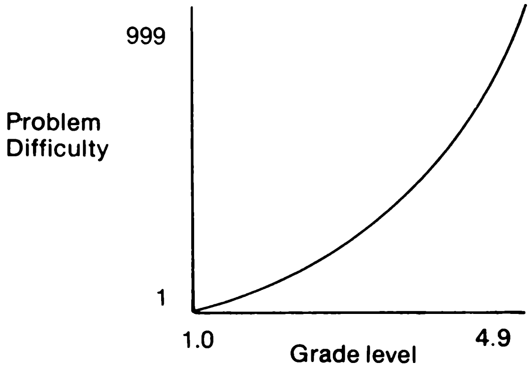
A grade level must be kept for the student for each type of problem that he is doing. So if a student is dealing with 9 different math concepts (or standards), he is assigned a grade level for each one.

Our next task is to assign a grade level to each problem presented. This unfortunately will vary depending upon the local school system, the textbooks used, and teaching method. Since we certainly don't want to store a huge data base of problems tagged with a grade level, for each problem type, we probably should try to devise a simple method of determining its grade level. We must also determine what level should be presented to the student. One straightforward approach is to present problems up to one-half a grade level over and under where the student currently is. Thus the overall range of problems for a student at grade level 3.2 would be 2.7 to 3.7.

How do we generate the right problems? Let's consider one type for now: vertical addition. Say it's first introduced in Grade 1 and continues through Grade 4 (actually 4.9). The simplest problem is 1 + 1 and most difficult 999 + 999. In other words:

Grade Level	*Addend*
1.0	1
4.9	999

Clearly, learning is not a linear process so we can't use a simple linear formula, hence we need something that represents the exponential process of learning.



There are lots of exponential representations like logs, powers, etc. A. fairly trivial, but workable formula for this problem is:

$$\text{Addend} = 1.73 \times \text{Grade Level}^4$$

or

$$\text{Grade level} = \sqrt[4]{\text{Addend}/1.73}$$

To fill in a few more values on our tables:

Grade level	*Addend*
1.0	1
2.0	27
3.0	140
4.0	442
4.9	997

Now it's a relatively straightforward, although somewhat tedious matter, to tie all these elements together in a computer program.

A few notes about the program. G2 is a variable that is within one-half a grade level of the current level of the student, G1. The complicated mess in statement 350 determines a fair approximation of grade level based on the answer to the problem. The recording of the grade level and carrying over to the next lesson is a manual process; grade level could just as easily be retained on a file medium such as floppy disk or cassette tape and keyed to the name or number (heaven forbid) of the student. ■

---

cosmic rays are bombarding my body

Cosmic rays
are bombarding
my body, streaking
through my flesh at
relativistic speeds,
yet I sit here fearlessly,
watching the Super Bowl on t.v.

Steelers 21
Cowboys 17

Peter Payack

Poems by Peter Payack have appeared in *The Paris Review*, *The Village Voice*, *Darkhorse*, and *Star-Web Paper*. A collection of 14 of Peter's poems are available in the booklet "Cornucopia" available for 70¢ from Peter Payack, 23 7th St., Cambridge, MA 02141.

FRIEND

There is a computer I would like to call friend.
Then all of my troubles and worries would end.
It is the lottery computer run by our state.
Millionaires are made by one stroke of fate.

If I were to befriend this electronic brain
My tastes again would run to champagne.
And the machine I would not soon forget,
For the two of us together make a fine duet.

I would make it some flowers made up of wire,
Feed it a voltage just a little bit higher.
Install light bulbs a tiny bit brighter,
Ease the tape tension a little bit slighter.

We millionaires are a bit touched in the head.
For my honorable intentions word now I spread,
Look soon for your newspaper story to carry
Headlines of "Computer and man soon to marry."

Edward Stewart

CAI: Further Considerations for Presenting Multiple-Problem Types

Part three in this five-part series on Computer-Assisted Instruction looks at incorporating a sliding grade level into programs that generate several types of problems.

Laura L. McLaughlin

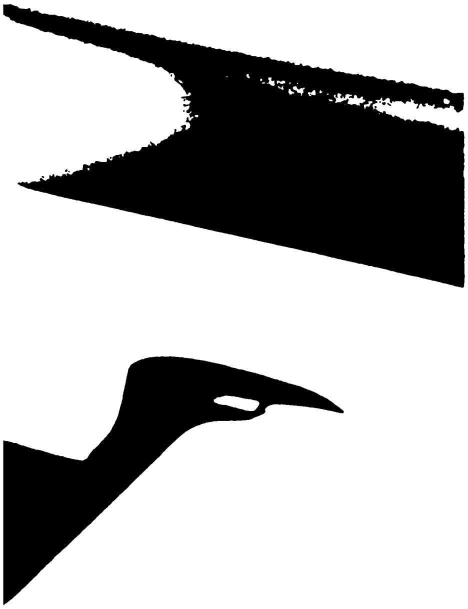


Photo by Lisa Sheble

```
B>RUN24 MATHART
BASIC-E INTERPRETER  VER K1 3

HI! WHAT'S YOUR NAME? TRACY
I GUESS WE'VE NEVER MET BEFORE TRACY
WHAT GRADE ARE YOU IN? 2

OKAY, LET'S PLAY WITH SOME NUMBERS

   1
 + 4
------
? 5

THAT'S RIGHT, TRACY

LET'S TRY ANOTHER

   21
 + 15
------
? 36

THAT'S RIGHT, TRACY

LET'S TRY ANOTHER

   41
 + 15
------
? 50

WRONG, TRY AGAIN? 55

WRONG, TRY AGAIN? 56

THAT'S RIGHT, TRACY

LET'S TRY ANOTHER..

   24
 + 14
------
? 38

THAT'S RIGHT, TRACY

LET'S TRY ANOTHER.

   54
 + 12
------
? 64

WRONG, TRY AGAIN? 54

WRONG, TRY AGAIN? 62

YOU DON'T SEEM TO UNDERSTAND THIS ONE TRACY
THE CORRECT ANSWER IS 66

LET'S TRY ANOTHER

41 + 7 =
```

EXAMPLE 1

```
10      DEF FNIN$(X)=LEFT$(STR$(X+.5),INT(LOG(X)/2.302585+1))
        DIM G(6)
        NAMES$="NAME.RND"
        NAMES=1
        FILE NAMES$(128)
        IF END #NAMES THEN 300
        INPUT "HI! WHAT'S YOUR NAME",NAME$
        RANDOMIZE
100     COUNT=COUNT+1
        READ #NAMES, NAMEF$, G(1), G(2), G(3), G(4), G(5), G(6)
        IF NAME$=NAMEF$ THEN 500
        GOTO 100
300     PRINT "I GUESS WE'VE NEVER MET BEFORE ",NAME$
        INPUT "WHAT GRADE ARE YOU IN",G(1)
        G(1)=G(1)+.3
        FOR I=1 TO 5
        G(I+1) = G(I)
        NEXT I
500     PRINT
        PRINT "OKAY, LET'S PLAY WITH SOME NUMBERS"


7000    PRINT
        PRINT "THAT'S ENOUGH FOR NOW ",NAME$
        PRINT "COME BACK AGAIN SOON"
        PRINT #NAMES, COUNT, NAME$, G(1), G(2), G(3), G(4), G(5), G(6)
9999    END

A>
```

The previous article in this series discussed a method of calculating a sliding grade level based on the range of the numbers presented in a particular problem. But how can we incorporate this concept into a program that handles multiple-problem types and what other factors should we consider?

Let's say we want a program that will generate six types of problems for grades 1-4: addition, subtraction, and multiplication in both the vertical and horizontal formats. We want this program to keep track of a student's grade level for each type of problem separately. We can then give him practice problems, in each area, that fall within a half-grade on either side of his current level for that particular type of problem. The program should constantly adjust the student's grade levels based on the accuracy of his responses.

We would also like the program to be both flexible and concise. Flexibility is important so that we can later expand and/or modify the types of problems to be presented or the grade levels to be covered. The need for conciseness is evident to all (except maybe those with their own personal IBM 370). To accomplish this, we are going to make extensive use of the BASIC FOR/NEXT and GOSUB verbs.

But before we can generate any problems, we must have a means of obtaining the student's grade levels. Since we are now talking about six different levels based on problem type and format, it is no longer practical to expect the student to enter his grade level data. Therefore, we are going to build a file with one record for each student. It will contain his name and a level for each type of problem. Example 1 shows the beginning and end of our program.

(1) At 100, we search through the file for the student's name. When a match is found we can proceed with the session, since we have his current levels.

(2) If there is no record for a student, we must ask him what grade he is in. This will provide us with initial values (see Line 300).

(3) Since his true level is probably somewhere between what he answers and the next higher grade, we will set our initial values up by three-tenths. Remember, the program will adjust these numbers in either direction based on his ability, so the initial grade levels are not absolute.

(4) The end of the program (at 7000) either updates or creates his record with the levels that have been adjusted during this session.

Just a word about the function (FNIN$) defined at the beginning of the program. As you may know, each of the different BASICs available has its own "quirks" (or, more accurately, problems). This program was developed using BASIC-E Version K1.4. One of its "quirks" is that on occasion it will incorrectly convert a numeric integer into a string variable as a non-integer number (for example, 93 becomes "93.00001"). The function FNIN$ is used to correct this problem. When writing a CAI program, it is essential that all such inaccuracies be identified and corrected or it will result in confusing, rather than helping, the student. Check your BASIC, nothing is perfect.

Now, what kinds of common subroutines can we use? Well, Example 2 shows four of the more obvious ones. The routine at 4000 will set up to three random numbers based on the value of G1 (the student's grade level). The formula for determining the maximum value of any one of the numbers is the same as used in the previous article for addition problems (Addend = 1.73 x Grade Level^4.)* However, we have provided the formula with greater flexibility by making the values for both FACTOR and POWER variable. This way they may be changed based on problem type. Note that the value of NUM must have been previously set for both this routine and the two that follow. Routines for printing a problem vertically or horizontally are shown at 5000 and 5500 respectively, with the type of problem determined by the value of OP$. The last routine, at 6000, will get the student's answer and process it. Three incorrect responses are accepted prior to showing the right answer. Note that this could be easily changed to vary depending on grade level, if we later determined that allowing only two mistakes would be more appropriate for the younger student. We then decide if the problem should effect his grade level and do the calculations if necessary.

We are now ready to write that part of the program which will actually generate the problems. The code in Example 3 will generate addition (1000), subtraction (2000), and multiplication (3000) problems. Let's look at it a little more closely and see what it really does:

(1) The values for G1 (student's grade level), OP$ (type of problem), FACTOR and POWER (parameters for calculating variables) are set based on the type of problem.

(2) Five problems of each type are presented in succession (with the exception that no multiplications are generated if his grade level for horizontal multiplication is below 2.5). By presenting the same math concept multiple times in succession, while at the same time limiting the number, we will give him the opportunity to learn from his mistakes but decrease the possibility of boredom or

EXAMPLE 2

```
4000     G2=G1-. 5+RND
         MAX=INT(FACTOR*G2^POWER)
         A=INT(RND*MAX)
         B=INT(RND*MAX)
         IF NUM=2 THEN C=0:RETURN
         C=INT(RND*MAX)
         RETURN


5000     PRINT
         PRINT "    ",
         IF A=0 THEN PRINT A ELSE PRINT FNIN$(A)
         IF NUM=2THEN 5100
         PRINT "    ",
         IF C=0 THEN PRINT C ELSE PRINT FNIN$(C)
5100     PRINT " ",OP$," ",
         IF B=0 THEN PRINT B ELSE PRINT FNIN$(B)
         PRINT "------"
         RETURN


5500     PRINT
         IF A=0 THEN PRINT A, ELSE PRINT FNIN$(A),
         PRINT\" ",OP$," ",
         IF NUM = 2 THEN 5600
         IF C=0 THEN PRINT C, ELSE PRINT FNIN$(C),
         PRINT " ",OP$," ".
5600     IF B=0 THEN PRINT B ELSE PRINT FNIN$(B),
         PRINT " = "
         RETURN


6000     INPUT ANSWER
         IF ANSWER = R THEN 6200
         W=W+1
         IF W>2 THEN 6100
         PRINT
         PRINT "WRONG, TRY AGAIN",
         GOTO 6000
6100     PRINT
         PRINT "YOU DON'T SEEM TO UNDERSTAND THIS ONE ",NAME$
         PRINT "THE CORRECT ANSWER IS ",
         IF R=0 THEN PRINT R ELSE PRINT FNIN$(R)
         IF G2>G1 THEN GOTO 6300 ELSE GOTO 6250
6200     PRINT
         PRINT "THAT'S RIGHT, ",NAME$
         IF G2<G1 THEN 6300
6250     MAX=SQR(MAX/FACTOR)
         IF POWER = 4 THEN MAX=SQR(MAX)
         G1= 9*G1+ 1*MAX
6300     W=0
         IF OP$="-" AND G(5)<2 5 AND I=10 THEN RETURN
         IF OP$="X" AND I=10 THEN RETURN
         PRINT
         PRINT "LET'S TRY ANOTHER    "
         RETURN

A>
```

---

*Some terminals use this "upside-down saucer" to indicate exponentiation, rather than an up-arrow.

EXAMPLE 3

```
1000    G1 = G(1)
        OP$ = "+"
        FACTOR = 1 73 POWER = 4
        FOR I = 1 TO 10
        IF G1<4 THEN NUM=2 ELSE NUM=3
        GOSUB 4000
        IF G2<2 5 AND I<6 THEN GOSUB 4500
        IF G2<3 AND I>5 THEN GOSUB 4500
        R=A+B+C
        IF I<6 THEN GOSUB 5000 ELSE GOSUB 5500
        GOSUB 6000
        IF I=5 THEN G(1)=G1 G1=G(2)
        NEXT I
        G(2)=G1
        GOSUB 6500

2000    G1=G(3)
        OP$="-" NUM=2
        FOR I=1 TO 10
        GOSUB 4000
        IF A<B THEN X=A A=B.B=X
        IF G2<3 AND I<6 THEN GOSUB 4500
        IF G2<3 5 AND I>5 THEN GOSUB 4500
        R=A-B
        IF I<6 THEN GOSUB 5000 ELSE GOSUB 5500
        GOSUB 6000
        IF I=5 THEN G(3)=G1 G1=G(4)
        NEXT I
        G(4)=G1
        GOSUB 6500

3000    G1=G(5)
        IF G1<2 5 THEN 7000
        OP$="X"
        FACTOR= 68 POWER=2
        FOR I=1 TO 10
        GOSUB 4000
        R=A*B
        IF I<6 THEN GOSUB 5000 ELSE GOSUB 5500
        GOSUB 6000
        IF I=5 THEN G(5)=G1 G1=G(6)
        NEXT I
        G(6)=G1
        GOTO 7000

A>
```

frustration.

(3) The value of NUM (number of variables) is set for addition based on the grade level of the problem (G2), while for subtraction and multiplication it is set to 2.

(4) When done with a particular problem type, the student's grade level for that type is updated.

We could have put the generation of all three problem types into one large FOR-NEXT loop. This approach would have saved us close to 50% of the code necessary to produce the problems, but would have made modifications and/or additions much more difficult (the old ease of maintenance/size & efficiency trade-off). The choice of coding technique should be based upon the extent of modification envisioned to achieve the final iteration.

The addition and subtraction loops also call two routines we have not yet defined. Take a look at Example 4. The routine at 4500 is called from addition and subtraction only for problems of that below a particular level. It will insure that addition problems will not require a "carry" if both numbers are greater than 9, and that subtractions will not need a "borrow." This kind of special editing (independent of the range of the variables) must be considered for each type of math concept presented. Otherwise, we run the risk of completely frustrating the student by expecting him to attempt something which could well be far beyond his capabilities. Similarly, since we do not want to give multiplication too soon, the routine at 6500 is used to set the student's grade level to a value that will allow the generation of multiplication problems only after he has reached a particular level (grade 3) in one of the other types.

Now that we have a basis from which to work, it will be relatively easy to expand upon. Subroutines can be written for generating different formats (mixed operations, fractions) or more operations (division, square roots). Also, additional checks can be included to give a student a larger variety of problems and eliminate those which he has already mastered.

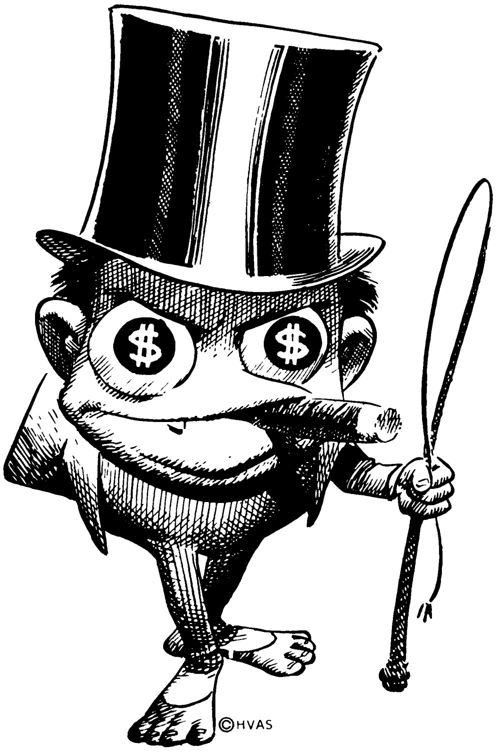
In the next article, we will discuss some of the considerations that should be made concerning the interaction between the student and the computer. Careful thought must be given to such things as edit requirements and presentation formats so that the student does not need to learn a whole new set of rules. Remember, for CAI to be a useful tool, it must be something that is easy and comfortable to use. ■

EXAMPLE 4

```
4500    IF A<10 OR B<10 THEN RETURN
        TSTA$=RIGHT$(FNIN$(A),1)
        TSTB$=RIGHT$(FNIN$(B),1)
        TSTA=VAL(TSTA$)
        TSTB=VAL(TSTB$)
        IF OP$="-" THEN 4600
        IF TSTA + TSTB < 10 THEN RETURN
        IF TSTA > 4 THEN A=A-5
        IF TSTB > 5 THEN B=B-5
        RETURN
4600    IF TSTA - TSTB >= 0 THEN RETURN
        IF TSTA < 5 THEN A=A+5
        IF TSTB > 5 THEN B=B-5
        RETURN

6500    IF G(5) >= 3 THEN RETURN
        IF G1<3 THEN RETURN
        G(5)=3
        G(6)=3
        RETURN

A>
```

"How did things go at the lab today Dear?"

Keeping the Loan Arranger Honest



James A. Warden*

Most of us have borrowed money at some time, whether to buy a house, to pay for a car or a large appliance, or to extend payment on a revolving charge account. The truth-in-lending laws may have made us aware that loans do cost money and that we can shop around for a place to borrow, but the details of computing the monthly payment schedule may still seem a bit mysterious. Yet these calculations are in fact quite easy and anyone with access to a computer (or even a calculator) can crank out loan schedules at will.

Let's get a few basic facts straight first. Usually we borrow an amount of money known as the principal amount, which is to be paid back in monthly installments. We must pay interest on the loan each month, which is based on an annual percentage of the principal, or the annual interest rate. We might wish to know answers to these questions about the loan: If I want to pay it back in N months, what is the monthly payment? If I want to pay back so much per month, how long will it take to repay it? What is the total amount I will pay?

There are standard loan formulas which can provide the answers, and these formulas really are not difficult to use. but you can't generate a payment schedule with them, and you may not wish to bother with the mathematics necessary to derive them. Instead, we can consider a simple recursive procedure which will provide all the answers to allow us to see what is going on at the same time. If we borrow an amount P and pay it back monthly at an annual interest rate of R (per cent), our monthly payment M is used first to pay the interest which has accrued on the principal during the month. Expressed mathematically, the interest due on the principal remaining is I = P*R/1200. The 1200 comes in because we must divide the annual rate by 12 to get the monthly rate, and we must divide the percentage by 100 to obtain a fraction. The monthly payment reduces the principal by M-I, leaving P+I-M to be repaid. Expressing this in the form of a "recursive relation", we have that P(new) = P+I-M, which tells us how much we have left to pay (the new principal amount) after making a payment on the old P. To find out how much we will owe next month, we take P(new), insert it in place of P, and repeat the calculation. Eventually, P(new) will be reduced to nothing, and we can "burn the note". Of course, P(new) may turn up negative on a particular payment, meaning that the payment M will overpay the loan. In this case, the final payment will be whatever principal is left plus the interest owed on it, or P+I. This algorithm is illustrated in flowchart form below.

To find out how many months it will take to repay the loan, we simply have to choose a monthly payment, start with the principal amount owed, compute the new principal owed after a month, and repeat the operation until the principal drops to zero, printing the result each cycle. If we also put a counter in the loop and display payment numbers, the number of payments will be obvious. Finding a monthly payment which pays back the loan in exactly N payments requires a bit of guesswork. Here we pick a reasonable payment, run the calculation until the principal drops to zero, note the number of months required to pay, and then adjust our guess of the monthly payment to come out closer to the desired N on the next calculation. In either case, simply summing the monthly payments made will give the total cost of the loan. That's all there is to the technique! If it still sounds a bit vague, study the flowchart. This algorithm can be written up in BASIC, FORTRAN or a calculator procedure.

If you would like some practice or you wish to introduce this technique to someone else, the BASIC program LOANER may be of help. This program is one of a series of routines used at Wabash College to generate random exercises for the elementary computer science classes. LOANER will make up loan problems at random and use standard loan formulas to estimate values for number of payments for a given monthly payment and the total amount paid.

I would like to thank Prof. T. Mielke of the Wabash Mathematics Department for some of the ideas developed in this article. ■

---

* Cragwall Computer Center Wabash College Crawfordsville, Indiana 47933

PROSPERITY

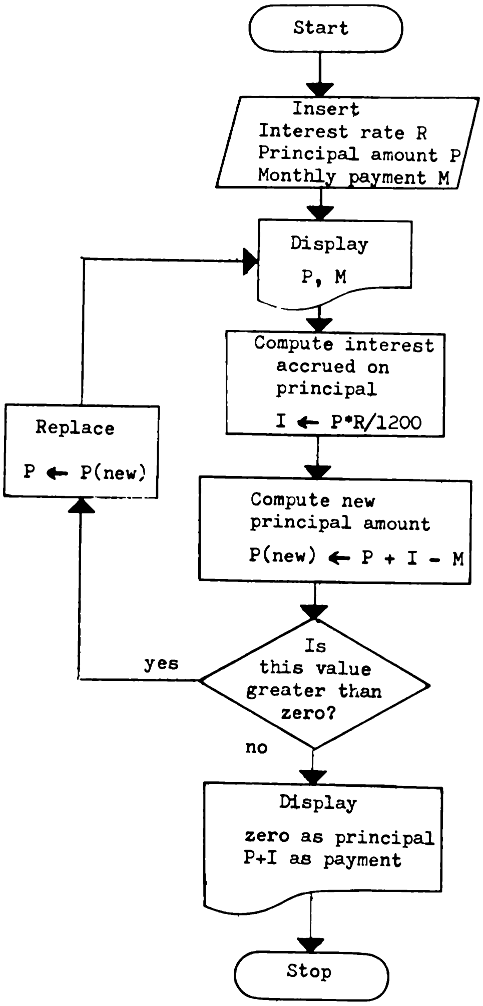
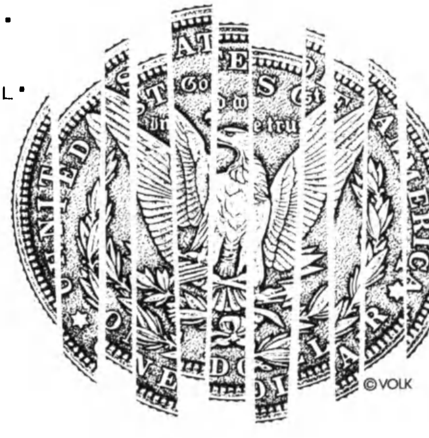

Start
Insert
Interest rate R
Principal amount P
Monthly payment M
Display
P, M
Compute interest
accrued on
principal
I ← P*R/1200
Compute new
principal amount
P(new) ← P + I - M
Is
this value
greater than
zero?
yes
no
Replace
P ← P(new)
Display
zero as principal
P+I as payment
Stop

Fig. 1 -- Flowchart for an algorithm which generates an amortization schedule

```
100 REM LOANER -- AN EXERCISE GENERATOR
120 REM J. WARDEN        WABASH COLLEGE
140     RANDOMIZE
160 REM GENERATE VALUES OF PRINCIPAL, RATE, AND PAYMENT
180     P=INT(RND*500)*10+1000.
200     R=INT(RND*20+20)*.5
220     M=INT(P*(.07+RND*.05))
240 PRINT
 260 PRINT"WRITE A BASIC PROGRAM WHICH WILL GENERATE AN AMORTIZATION"
280 PRINT"SCHEDULE OF MONTHLY PAYMENTS FOR A LOAN OF  $";P
300 PRINT"TO BE REPAID AT AN ANNUAL INTEREST RATE OF";R;"PER CENT."
320 PRINT"ASSUME THAT THE MONTHLY PAYMENT IS  $";M" ."
340 PRINT
 360 PRINT"INCLUDE IN THE SCHEDULE THE PAYMENT NUMBER, THE PRINCIPAL"
380 PRINT"REMAINING, INTEREST, AMOUNT PAID TO PRINCIPAL, AND THE"
400 PRINT"PAYMENT.  AT THE END, DISPLAY THE TOTAL AMOUNT PAID."
420 REM GENERATE COMPARISON VALUES FOR NUMBER OF PAYMENTS
440 REM AND TOTAL AMOUNT PAID FOR THE LOAN
460     A = 1 + R/1200
480     N=INT( (LOG(M)-LOG(M-P*(A-1.)))/LOG(A) )
500     A2=(1.-A^N)/(1-A)
520     T=P+(R/1200)*(P*A2 - M*(N-A2)/(1-A))
540     T=INT(T+.49)
560 PRINT
 580 PRINT"(HINT: FOR THE FIGURES GIVEN, THE LOAN SHOULD REQUIRE";N
600 PRINT"PAYMENTS AND A TOTAL OF ABOUT  $";T;"WILL BE PAID.)"
620 END

 READY
```



short programs.....

Changing Bases

Here's a clever little program (which could be used as a subroutine in a larger program) to convert from one base to another, in particular from base 10 to a new base between 2 and 16. The crucial parts of the routine are found in statements 150-170 which determine how many digits are in the new number by raising the new base to successively higher powers and checking if the resultant number is larger than that to be converted. For example, if we wanted to convert 15 to base 2, statements 150-170 would change variables as follows to finally meet the condition of B^P greater than N(1).

B	P	B^P	N(1)
2	0	1	15
2	1	2	15
2	2	4	15
2	3	8	15
2	4	16	15

Hence, the routine has determined that there are 4 digits (P) in the new number.

Statements 180-210 then do the actual conversion and deposit the new number in D(1) through D(P). We're not going to tell you how it works but leave that up to you to figure out.

Thanks to Jim West of Teletype Corp. for leaving this little gem in my GE timesharing account. — DHA.

Program Listing

```
090 REM   BY JIM WEST, WILMETTE, IL, (312)256-1621, JAN 1977
095 DIM D(20), N(20)
096 FØR J=1 TØ 6\ READ D$(J)\ NEXT J
097 DATA "A","B","C","D","E","F"
100 PRINT "THIS PRØGRAM WILL TAKE A PØSITIVE INTEGER (BASE 10)"
110 PRINT "AND CØNVERT IT TØ ANY NEW BASE BETWEEN 2 AND 16"\PRINT
115 REM       INPUT
120 PRINT "NEW BASE =";\INPUT B
125 IF B < 2 THEN120
126 IF B > 16 THEN 120
127 IF B - INT (B) > 0 THEN 120
130 PRINT "INTEGER (BASE 10) =";\INPUT N(1)
131 IF N(1)-INT(N(1)) > 0 THEN 130
133 IF N(1) < 0 THEN 130
135 REM     CALCULATIØN
140 LET P = 0
 150 IF B^P > N(1) THEN 180
160 LET P = P + 1
170 GØTØ 150
180 FØR I = 1 TØ P
190 D(I) = INT (N(I)/B^(P-I))
200 LET N(I+1) = N(I) - (D(I)*B^(P-I))
210 NEXT I
220 REM       ØUTPUT
230 PRINT\PRINT
235 PRINT N(1); "(BASE 10) =";
240 FØR I = 1 TØ P
245 IF D(I) > 9 THEN 255
250 PRINT D(I);\GØTØ 260
255 PRINT D$(D(I)-9);
260 NEXT I
265 PRINT " (BASE"; B; ")"
270 PRINT\PRINT\PRINT
300 GØTØ 120
999 END
```

The input section checks for invalid input, decimals, negative numbers and the like

The digits of the new number are put in D(1) - D(P).

Sample Run

```
THIS PRØGRAM WILL TAKE A PØSITIVE INTEGER (BASE 10)
AND CØNVERT IT TØ ANY NEW BASE BETWEEN 2 AND 16

NEW BASE =? 2
INTEGER (BASE 10) =? 1977


 1977 (BASE 10) = 1  1  1  1  0  1  1  1  0  0  1  (BASE 2 )


NEW BASE =? 8
INTEGER (BASE 10) =? 1977


 1977 (BASE 10) = 3  6  7  1  (BASE 8 )


NEW BASE =? 12
INTEGER (BASE 10) =? 1977


 1977 (BASE 10) = 1  1  8  9  (BASE 12 )


NEW BASE =? 16
INTEGER (BASE 10) =? 1977


 1977 (BASE 10) = 7  B  9  (BASE 16 )
```

· FOR BOOKS & SUBSCRIPTIONS ·
800
631-8112
IN NJ CALL
201-540
-0445
BANKAMERICARD · MASTERCHARGE

Delving Into Depreciation

This simple little program computes annual depreciation on a capital item using 3 different types of depreciation. The IRS generally allows any type of depreciation as long as it is consistent; in other words you can't start depreciating an item using the double declining balance method and then switch to straight line.

Why different methods of depreciation? Because depreciation serves different purposes. Say you're a car leasing firm; then most of the depreciation occurs early in the life of your capital goods (automobiles) and you might want to use the double declining balance or sum of digits method to reflect this. On the other hand, an auto service shop might buy a hoist with an expected 5-year life. Because you expect the income of the shop to increase over time you'd like to push as much of your expenses as possible to future years to offset the increased taxes. Hence, you would opt for straight-line depreciation.

To delve further into the mysteries of depreciation as an indirect source of capital and also the tax effects, I recommend almost any business finance text such as Hunt, Williams, and Donaldson: *Basic Business Finance* published by Irwin. — DHA

```
120 PRINT "ØRIGINAL CØST ØF ITEM (DØLLARS, NØ CENTS)";
130 INPUT C
140 PRINT "LIFE ØF ITEM (YEARS, NØ MØNTHS)";
150 INPUT L
160 PRINT "SCRAP VALUE (DØLLARS, NØ CENTS)";
170 INPUT S
180 PRINT
190 PRINT "YEAR","STRAIGHT","SUM ØF"," DØUBLE"
200 PRINT " ","  LINE","DIGITS","DECLINING"
210 PRINT
220 V=C-S
230 D1=V/L
240 Y=((L+1)/2)*L
250 Z=L
260 FØR X=1 TØ L
270 D2=V*(Z/Y)
280 Z=Z-1
290 D3=2*C/L
300 C=C-D3
310 PRINT X,
312 Q=D1
314 GØSUB 400
 316 Q=D2
318 GØSUB 400
 320 Q=D3
322 GØSUB 400
 325 PRINT
330 NEXT X
350 PRINT "FINISHED"
360 GØTØ 999
400 Q=INT(Q*100)/100
420 IF Q>100 THEN 440
430 PRINT " ";
440 IF Q>10 THEN 460
450 PRINT " ";
 460 PRINT "$"Q,
490 RETURN
999 END
```

Value to be depreciated = Cost - Scrap Value

Straight Line dep'n = Value / Years

Sum of digits depreciation

Double declining balance depreciation

Print routine rounds off to an even number of dollars or dollars and cents.

```
RUN

ØRIGINAL CØST ØF ITEM (DØLLARS, NØ CENTS)? 3900
LIFE ØF ITEM (YEARS, NØ MØNTHS)? 5
SCRAP VALUE (DØLLARS, NØ CENTS)? 400

YEAR            STRAIGHT        SUM ØF          DØUBLE
                  LINE          DIGITS          DECLINING

 1              $ 700           $ 1166.66       $ 1560
 2              $ 700           $ 933.33        $ 936
 3              $ 700           $ 700           $ 561.59
 4              $ 700           $ 466.66        $ 336.95
 5              $ 700           $ 233.33        $ 202.17
FINISHED
```

Systematic Savings

When is it worth taking a simple algebra or finance or engineering formula and writing a computer program around it? Why not simply use an $8 calculator to get the result? Generally that's the best bet except when you want to use the formula over and over again or when amounts are being accumulated from one calculation to the next one.

Here's such an example. This program computes the total amount of money which will accumulate under a systematic investment program. The entire forumla is contained in Line 20; S is the net investment year by year. Use it to find out how much you could save if you weren't investing in a home computer, peripherals, terminals, software, etc., etc. — DHA

```
1 PRINT "THIS PRØGRAM CØMPUTES THE TØTAL AMØUNT ØF MØNEY"
2 PRINT "WHICH WILL ACCUMULATE UNDER A SYSTEMATIC INVESTMENT PRØGRAM"
3 PRINT\PRINT "HØW MUCH DØ YØU WANT TØ INVEST PER YEAR";
4 INPUT A\PRINT "HØW MANY YEARS";\INPUT N
5  PRINT "ANNUAL RATE ØF INTEREST (PERCENT)";\INPUT R
6 PRINT\PRINT "YEAR","TØTAL INVEST","TØTAL ACCUMULATED"
10 FØR N1=1 TØ N
20 S=A*(((1+(R/100))^N1)-1)/(R/100)
30 PRINT N1,N1*A,S
40 NEXT N1
99 END
```

The standard interest accumulation formula seems disguised in Basic code:

$$S = \frac{A[(1+r)^n - 1]}{r}$$

```
READY
RUN

THIS PRØGRAM CØMPUTES THE TØTAL AMØUNT ØF MØNEY
WHICH WILL ACCUMULATE UNDER A SYSTEMATIC INVESTMENT PRØGRAM

HØW MUCH DØ YØU WANT TØ INVEST PER YEAR? 5000
HØW MANY YEARS? 10
ANNUAL RATE ØF INTEREST (PERCENT)? 6.7

YEAR           TØTAL INVEST    TØTAL ACCUMULATED
 1              5000            5000
 2              10000           10335
 3              15000           16027.4
 4              20000           22101.3
 5              25000           28582.1
 6              30000           35497.1
 7              35000           42875.4
 8              40000           50748.
 9              45000           59148.1
 10             50000           68111.1
```

Those who deny freedom to others deserve it not for themselves.

Anamorphic Art

Andy A. Zucker

Pub. notes: Andy first sent us his program in BASIC-PLUS (for DEC PDP-11 systems). I wrote back and requested some explanatory notes for those readers not so fortunate to have BASIC-PLUS at their fingertips. Andy responded by not only explaining the BASIC-PLUS goodies he used, but then he translated his program to more-or-less "standard" BASIC. It is this translated program that you now see here. It probably should be noted that a colon (:) separates multiple statements on one line and an exclamation point (!) is equivalent to REMARK but may appear anyplace on a line. Once it appears, nothing else is executed on that line. Due to the use of a DECwriter for output, the exponentiation symbol, normally an up-arrow, looks like a funny umbrella without a shaft (my daughter calls it an upside-down saucer—see Line 580).

Andy also kindly enclosed a piece of reflectized (mirror finish) flexible Mylar, about 8" x 8", which can be rolled into a cylindrical mirror. This, of course, is crucial to correctly viewing cylindrical anamorphic distortions. I found that it's vital to have a smooth, mirror-like viewing surface. Smoothed-out aluminum foil just won't do—too much distortion from the inevitable crinkles. Wish we could have bound in a piece of reflecticized Mylar, but the cost was prohibitive. There currently is a Springbok anamorphic jigsaw puzzle on the market which is quite (very) challenging. It comes with a piece of mylar, although you can certainly find a piece cheaper in a local art-supplies shop or elsewhere. (If you're a puzzle freak like I am, you'll opt for the puzzle, of course.) My applause to Andy for a most original and creative application!—DHA.

Anamorphic (distorted) art has been the subject of a very popular international exhibit recently displayed at the Museum of Fine Arts in Boston. Martin ("Mathematical Games") Gardner wrote on this topic in the January 1975 issue of *Scientific American.* Also, several collections of anamorphic art have recently been published (such as *Hidden Images*, published by H.N. Abrams), and *Natural History* magazine recently had an article on the subject.

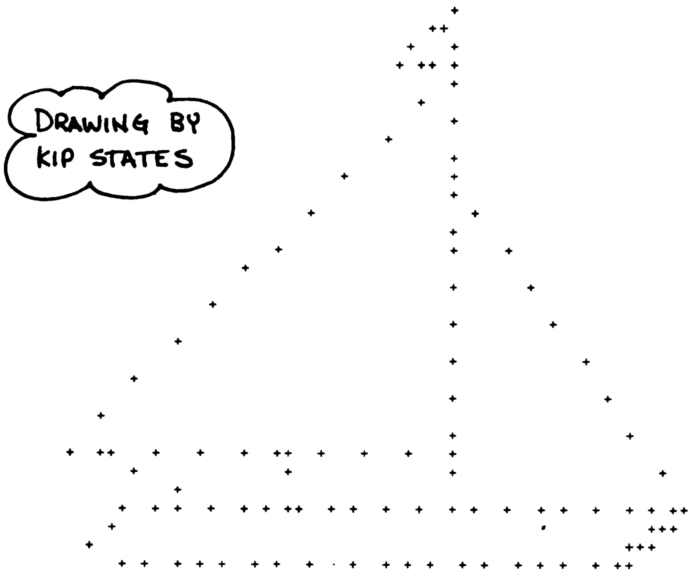
What I've done is to write a computer program which produces a certain, popular kind of anamorphic art—namely cylindrical distortions. These drawings look weird unless they are viewed in a cylindrical mirror placed vertically at the proper location.

To use the program to make a cylindrical anamorphism one begins by drawing a normal picture on a 0-to-60 (x) by 0-to-60 (y) grid, such as 10-square-per-inch graph paper. The picture is entered into the computer as a series of points, lines, circles, or portions of circles. The picture may be output "as is" to check its accuracy. Finally, the "transform" command causes the computer to compute the image of each picture-point, order the image-points for output, and draw the anamorphism.

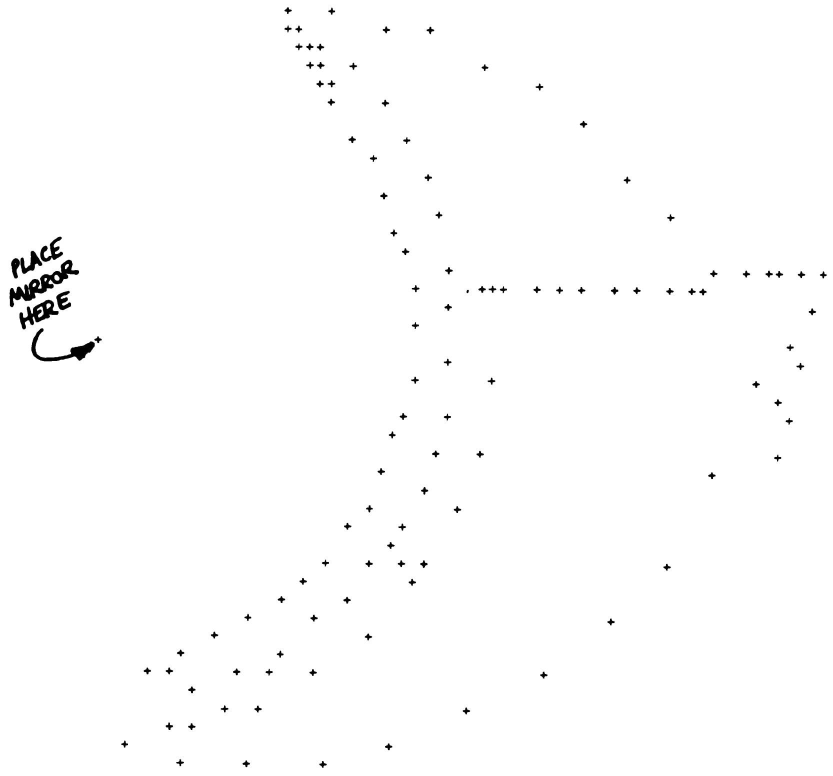
The results are surprising and amusing. They will interest artists, mathematicians, and anyone who is interested in the "graphics" capility of a simple terminal like a Teletype. Analyzing how the program works would make a good exercise for h.s. math students, as it involves polar coordinates, sines, and cosines.

Andy A. Zucker is on the faculty of Milton Academy, 170 Centre Street, Milton, Massachusetts 02186.

```
COMMAND? D
```



```
COMMAND? T
```



```
10      REM     ANAMORPHIC ART PROGRAM

20      REM     BY ANDY ZUCKER, MILTON ACADEMY, DECEMBER 1976
                 (WITH THANKS TO J. PIAZZA)

50      INPUT"INSTRUCTIONS (Y/N)";I$
        : IF I$<>"Y" THEN 200

54 PRINT: PRINT
55 PRINT TAB(4);"THIS PROGRAM WILL:":
 PRINT "A) ENTER A PICTURE FROM THE TERMINAL USING COMMANDS TO CREATE":
 PRINT "POINTS, LINES, CIRCLES, OR PORTIONS OF CIRCLES"
60 PRINT "B)  DRAW THE PICTURE ENTERED"
65 PRINT "C)  DRAW A DISTORTED PICTURE (CALLED AN ANAMORPHISM) WHICH":
 PRINT "LOOKS IDENTICAL TO B) IF IT IS VIEWED IN A 2.5 INCH DIAMETER":
 PRINT "CYLINDRICAL MIRROR (WHICH CAN BE MADE FROM SILVERED MYLAR PAPER)"
70 PRINT
75 PRINT TAB(4);"YOUR PICTURE MAY CONTAIN UP TO 200 POINTS":
 PRINT "THE PROGRAM ASSUMES A SQUARE PICTURE GRID WITH X AND Y":
 PRINT "COORDINATES FROM 0 TO 60 (NEED NOT BE INTEGERS)"
80 PRINT
85 PRINT TAB(4);"VALID COMMANDS ARE:":
 PRINT "P (OR POINTS)";TAB(20);"TO ENTER SOME POINTS":
 PRINT "L (OR LINE)";TAB(20);"TO ENTER A STRAIGHT LINE SEGMENT"
90 PRINT "C (OR CIRCLE)";TAB(20);"TO ENTER ALL OR PART OF A CIRCLE":
 PRINT"D (OR DRAW)";TAB(20);"TO VIEW THE (NORMAL) DRAWING":
 PRINT "T (OR TRANSFORM)";TAB(20);"TO PRODUCE THE ANAMORPHISM"
95 PRINT "E (OR EXIT)";TAB(20);"TO STOP THE PROGRAM":&:&


200     DIM X(201),Y(201)
210     DIM N(120)
310     PRINT "SO FAR";N;"POINTS ENTERED."
320     PRINT: INPUT "COMMAND";C$
325     IF (C$="DRAW" OR C$="D") THEN 2050 ELSE
        IF (C$="TRANSFORM" OR C$="T") THEN 1030
330     IF (C$="POINTS" OR C$="P") THEN 400 ELSE
        IF (C$="LINE" OR C$="L") THEN 450 ELSE
        IF (C$="CIRCLE" OR C$="C") THEN 500 ELSE
        IF (C$="EXIT" OR C$="E") THEN 9999 ELSE GOTO 320

400     INPUT "HOW MANY";P                    !!!   POINTS   !!!
410     INPUT "X,Y";X(I),Y(I) FOR I=N+1 TO N+P
        : N=N+P
420     GOTO 310

450     INPUT "ENDPOINT--X,Y";A(I);B(I) FOR I=1 TO 2 !!! LINES  !!!
460     INPUT"NO. OF POINTS ON THE LINE";P
        : IF P<2 THEN 460
470     S=1/(P-1)
480     FOR I=N+1 TO N+P
                :X(I)=(I-N-1)*S*[A(2)-A(1)]+A(1)
                :Y(I)=(I-N-1)*S*[B(2)-B(1)]+B(1)
490     NEXT I: N=N+P: GOTO 310

500     INPUT "COORDINATES OF CENTER";A(1);B(1)      !!! CIRCLES !!!
```

```
505     INPUT "NO. OF POINTS ON THE ARC";P
510     INPUT "ENTIRE CIRCLE (E) OR PORTION ONLY (P)";X$
        : IF X$="E" THEN INPUT "RADIUS";R ELSE GOTO 550
520     S=2*PI/P : T=0
530     FOR I=N+1 TO N+P
                :X(I)=R*COS[T +(I-N-1)*S] +A(1)
                :Y(I)=R*SIN[T +(I-N-1)*S] +B(1)
540     NEXT I: N=N+P: GOTO 310

550     PRINT "COORDINATES OF ARC START-POINT"          ! CIRCLES (CONT) !
555     INPUT "GOING COUNTER-CLOCKWISE";A(2);B(2)
560     INPUT "FRACTION OF CIRCLE (LESS THAN 1)";F
        : IF F>=1 THEN 560
570     IF A(2)-A(1)=0 THEN 575
572             T=ATN [ (B(2)-B(1))/(A(2)-A(1))]
                : IF A(2)<A(1) THEN T=T+PI
573             GOTO 580
575             T=PI/2 : IF B(2)<B(1) THEN T=-T
580     S=2*PI*F/(P-1) : R=SQR [(B(2)-B(1))^2 +(A(2)-A(1))^2 ]
        : GOTO 530


1000    REM THIS SECTION TRANSFORMS EACH POINT TO ITS IMAGE IN THE
                ANAMORPHISM
1030    LET K=.45 : L=-1.5        ! THETA =KX+L (THETA IN RADIANS)
1040    LET A=.75 : B=2.5         ! RADIUS = AY+B (RADIUS IN INCHES)
                                  ! (CONSTANTS DETERMINED EMPIRICALLY)
1050    FOR I=1 TO N    ! TRANSFORM EACH POINT
1060            X=.1*X(I) : Y=.1*Y(I)   ! X, Y IN INCHES
1070            T=K*X+L  : R=A*Y+B
1080            X(I)=10*R*COS(T)
                : Y(I)=10*R*SIN(T)      ! NOW X,Y IN TENTHS OF INCHES AGAIN
1090    NEXT I


2000    REM     THIS SECTION ORDERS THE POINTS IN THE ARRAY FOR DISPLAY
2010    N=N+1: X(N),Y(N)=0   ! SHOW THE ORIGIN IN AN ANAMORPHISM
2050    N(I)=0 FOR I=0 TO 120
2060    FOR A=1 TO N    !!!   SORT BY Y-COORDINATE !!!
2070            F=0
                : FOR B=1 TO N-A
2080                    IF Y(B)>=Y(B+1) THEN 2110
2090                    F=1
                        :T=Y(B) : T2=X(B)
2100                    Y(B)=Y(B+1) :X(B)=X(B+1)
                        :Y(B+1)=T : X(B+1)=T2
2110            NEXT B
2120            IF F=0 THEN 2140
2130    NEXT A

2140    R=1:N2=0: L=FNR(Y(1))
2150    FOR I=1 TO N            ! HOW MANY POINTS IN EACH ROW??
2160            IF FNR(Y(I))=L THEN N2=N2+1: GOTO 2190
2170            N(R)=N2
                : N2=1 : R=R+1
2180            L=FNR(Y(I))
```

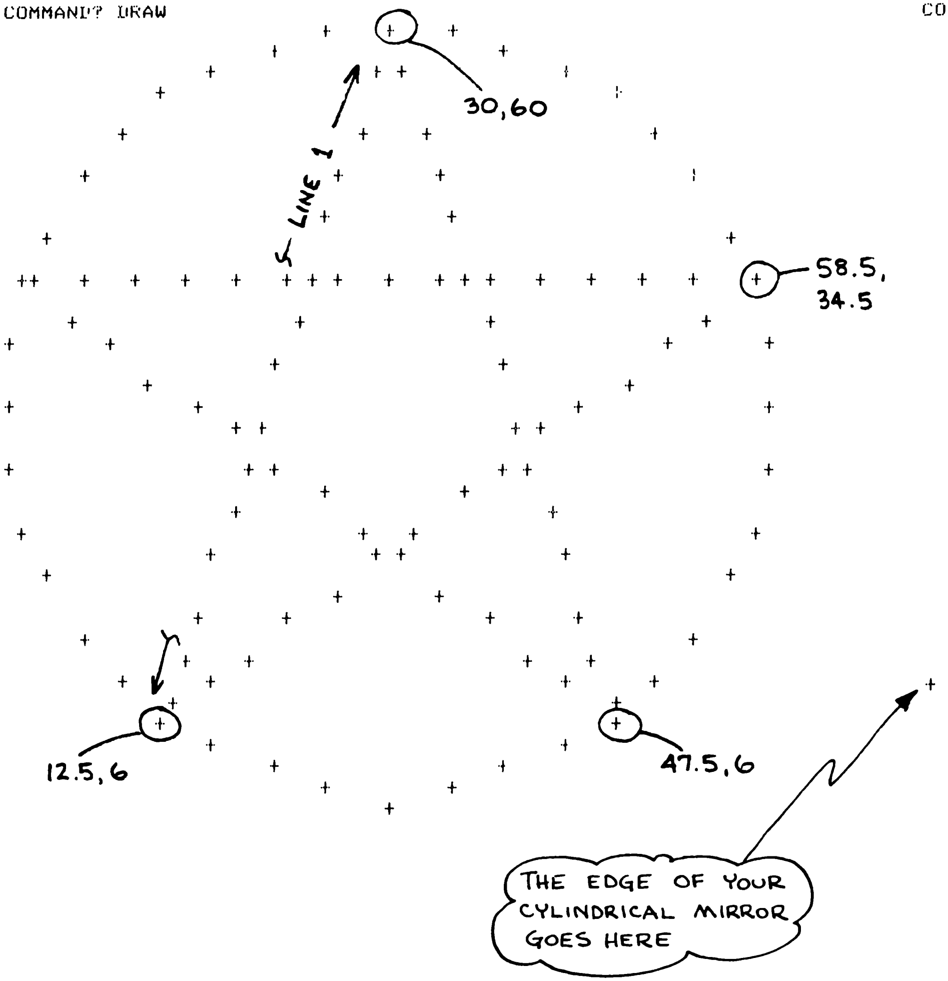
```
2190    NEXT I
2200    N(R)=N2

2210    N(I)=N(I)+N(I-1) FOR I=1 TO R  ! MAKE N() CUMULATIVE
2220    FOR I=1 TO R           ! FOR EACH ROW OF OUTPUT, SORT BY X-COORD.
2230            FOR A=1 TO N(I)-N(I-1): F=0
2240                    FOR B=N(I-1)+1 TO N(I)-A
2250                            IF X(B)<=X(B+1) THEN 2280
2260                            F=1
                                :T=Y(B) : T2=X(B)
2270                            Y(B)=Y(B+1) : X(B)=X(B+1)
                                : Y(B+1)=T : X(B+1)=T2
2280                    NEXT B
2290                    IF F=0 THEN 2310
2300            NEXT A
2310    NEXT I

3000    REM     THIS SECTION DRAWS THE PICTURE(S)
3010    C2$="+"         ! C2$ IS THE CHARACTER TO DRAW WITH
3020    S=0
3030    L=FNR(Y(1))
3040    FOR I=1 TO N            ! FOR EVERY POINT IN THE ORDERED FILE
3050            IF FNR(Y(I))<>L THEN
                        PRINT FOR J=1 TO L-FNR(Y(I))
                        : L=FNR(Y(I)) : S=0
3060            IF S>INT(X(I)+.5) THEN 3070
3065            PRINT TAB [INT(X(I)+.5)];C2$;
                : S= INT(X(I)+.5)+1  ! S GIVES POSITION OF PRINT HEAD
3070    NEXT I
3080    PRINT: IF (C$="DRAW" OR C$="D") THEN 320


3090    DEF FNR(Y)=INT( .6*Y+.5 )
                !  FUNCTION ADJUSTS FOR HORIZONTAL SPACING
                   AND ROUNDS OFF TO THE NEAREST LINE


9999    END
```

```
Ready

RUN ANAMOR
INSTRUCTIONS (Y/N)? Y


    THIS PROGRAM WILL:
A) ENTER A PICTURE FROM THE TERMINAL USING COMMANDS TO CREATE
POINTS, LINES, CIRCLES, OR PORTIONS OF CIRCLES
B)  DRAW THE PICTURE ENTERED
C)  DRAW A DISTORTED PICTURE (CALLED AN ANAMORPHISM) WHICH
LOOKS IDENTICAL TO B) IF IT IS VIEWED IN A 2.5 INCH DIAMETER
CYLINDRICAL MIRROR (WHICH CAN BE MADE FROM SILVERED MYLAR PAPER)

    YOUR PICTURE MAY CONTAIN UP TO 200 POINTS
THE PROGRAM ASSUMES A SQUARE PICTURE GRID WITH X AND Y
COORDINATES FROM 0 TO 60 (NEED NOT BE INTEGERS)

    VALID COMMANDS ARE:
P (OR POINTS)       TO ENTER SOME POINTS
L (OR LINE)         TO ENTER A STRAIGHT LINE SEGMENT
C (OR CIRCLE)       TO ENTER ALL OR PART OF A CIRCLE
D (OR DRAW)         TO VIEW THE (NORMAL) DRAWING
T (OR TRANSFORM)    TO PRODUCE THE ANAMORPHISM
E (OR EXIT)         TO STOP THE PROGRAM


SO FAR 0 POINTS ENTERED.

COMMAND? CIRCLE
COORDINATES OF CENTER? 30,30
NO. OF POINTS ON THE ARC? 40
ENTIRE CIRCLE (E) OR PORTION ONLY (P)? E
RADIUS? 30
SO FAR 40 POINTS ENTERED.

COMMAND? LINE
ENDPOINT--X,Y? 30,60
ENDPOINT--X,Y? 12.5,6
NO. OF POINTS ON THE LINE? 15
SO FAR 55 POINTS ENTERED.

COMMAND? LINE
ENDPOINT--X,Y? 30,60
ENDPOINT--X,Y? 47.5,6
NO. OF POINTS ON THE LINE? 15
SO FAR 70 POINTS ENTERED.

COMMAND? LINE
ENDPOINT--X,Y? 12.5,6
ENDPOINT--X,Y? 58.5,39.5
NO. OF POINTS ON THE LINE? 15
SO FAR 85 POINTS ENTERED.

COMMAND? LINE
ENDPOINT--X,Y? 47.5,6
ENDPOINT--X,Y? 1.5,39.5
NO. OF POINTS ON THE LINE? 15
SO FAR 100 POINTS ENTERED.

COMMAND? LINE
ENDPOINT--X,Y? 1.5,39.5
ENDPOINT--X,Y? 58.5,39.5
NO. OF POINTS ON THE LINE? 15
SO FAR 115 POINTS ENTERED.
```

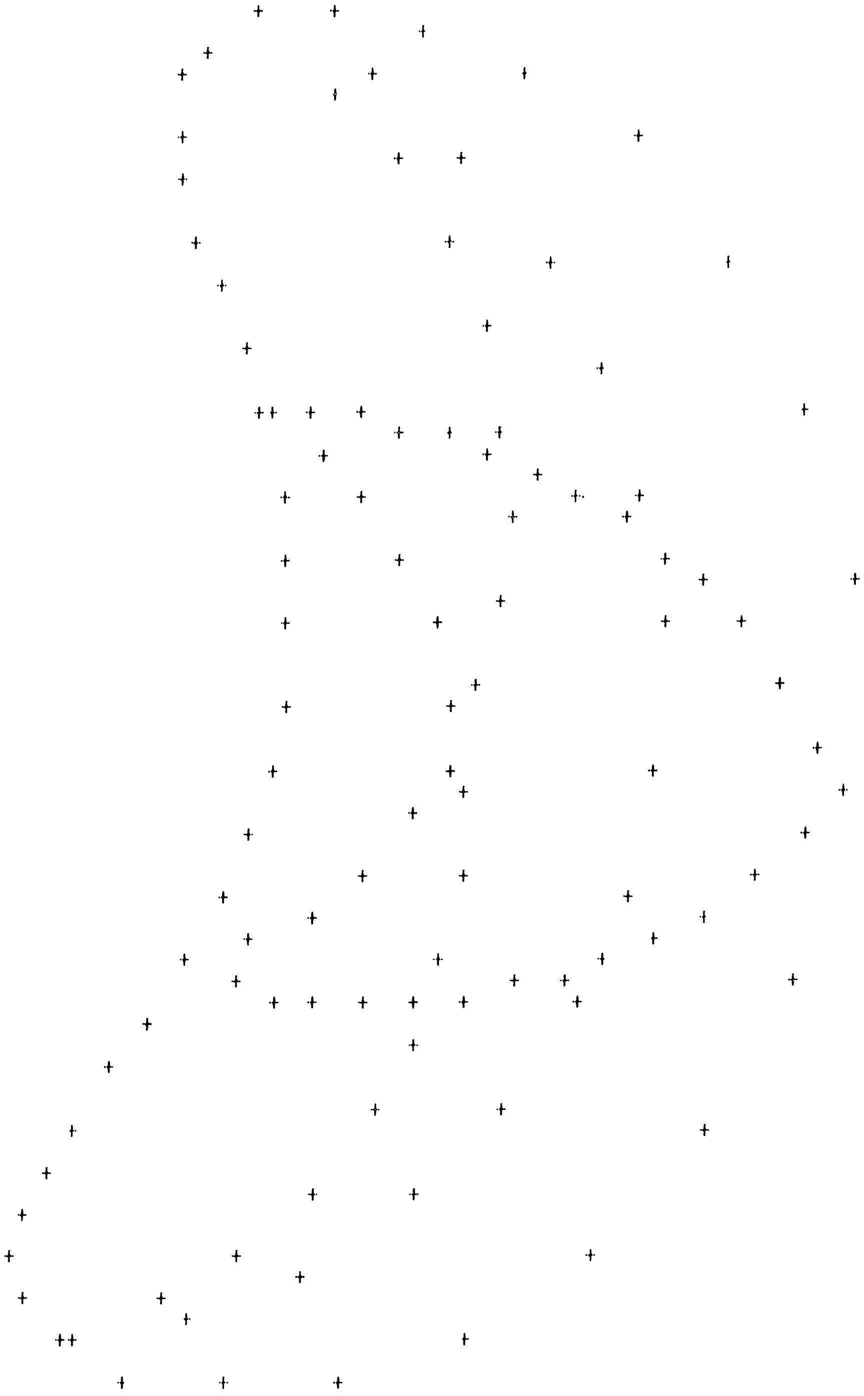
} THIS DRAWS LINE 1

TO DRAW A STAR
YOU HAVE TO FIGURE
OUT THE COORDINATES
OF THE FIVE POINTS.
IT HELPS TO DRAW
YOUR PICTURE FIRST
ON QUADRILLE OR
SQUARE GRAPH PAPER.

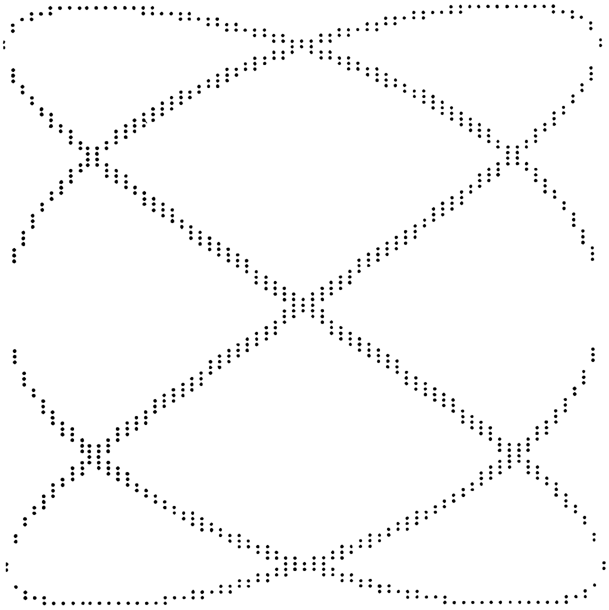
```
COMMAND? DRAW
```



```
COMMAND? TRANSFORM
```



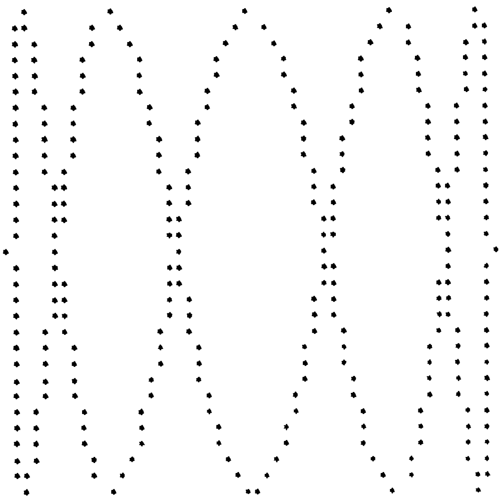
LISSAJOUS



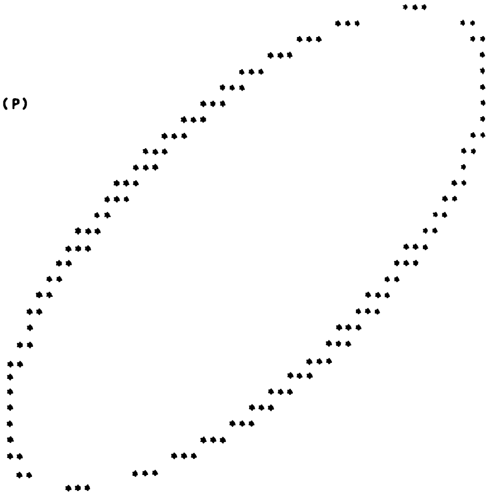
This program prints Lissajous patterns. You enter relative x and y frequencies, and phase. Two versions of the program are listed here—one in MITS Extended BASIC, and one in IBM 370 BASIC for more primitive machines. The IBM 370 version permits you to specify the width of the lines in the pattern. Note that the program in MITS BASIC uses the FIX function. If your BASIC doesn't have this, try Y1=SGN(Y1)*INT(ABS(Y1)+.5), etc.

The Lissajous pattern plotter was written by Larry Ruane and modified by several other people.

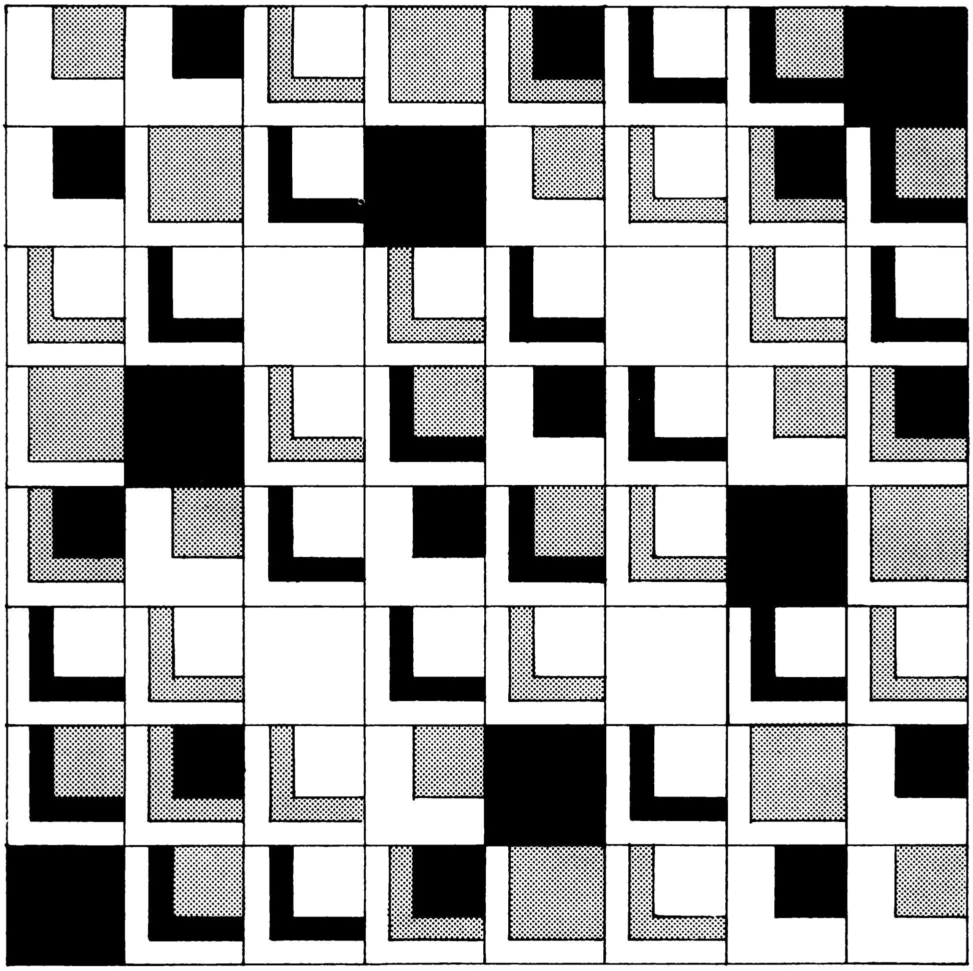
```
10 DIM Y(10)
100 REM.  STEP-VISE LISSAJOUS
110 PRINT:P=3.14159
120 PRINT "RELATIVE FREQ. FOR X";:INPUT F1:IF INT(F1)<F1 THEN 120
122 IF F1<1 THEN 120
125 F=F1:F1=2*P*F1
130 PRINT "RELATIVE FREQ. FOR Y";:INPUT F2: IF INT(F2)<F2 THEN 130
132 IF F2<1 THEN 130
135 PRINT "Y PHASE, MULTIPLE OF PI";:INPUT P2:P2=P*P2
140 F2=2*P*F2
150 FOR X1=-16 TO 16
160 X=X1/16:GOSUB 1970:T1=X:T2=P-X
162 FOR I=0 TO F-1
165 T3=(T1+2*I*P)/F1:T4=(T2+2*I*P)/F1
170 Y1=30*SIN(F2*T3+P2):Y2=30*SIN(F2*T4+P2)
180 Y1=FIX(Y1+SGN(Y1)/2):Y2=FIX(Y2+SGN(Y2)/2)
190 Y(2*I)=Y1:Y(2*I+1)=Y2
200 NEXT I
210 FOR J=1 TO 2*F-1:I=J-1:T=Y(J)
220 IF T>=Y(I) THEN 240
230 Y(I+1)=Y(I):I=I-1:IF I>=0 THEN 220
240 Y(I+1)=T:NEXT J
250 FOR I=0 TO 2*F-1
260 IF I=0 THEN 280
270 IF Y(I)=Y(I-1) THEN 290
280 PRINT TAB(36+Y(I));"*";
290 NEXT I
300 PRINT
310 NEXT X1
1890 STOP
1960 REM:---------------
1970 IF ABS(X)<.1 THEN 2020
1980 X=X/(SQR(1+X)+SQR(1-X))
1990 GOSUB 1970
2000 X=2*X
2010 RETURN
2020 X=X+X↑3/6+.075*X↑5+X↑7/22.4
2030 RETURN
2040 END
```



```
00010 REM - LISSAJOUS PATTERN PLOTTER  BY L. RUANE, ST. VIATOR H.S.
00020 REM - IBM 370 - HARPER COLLEGE, PALATINE, ILLINOIS
00030 DIM M(120)
00040 PRINT "ENTER FREQUENCY RATIO - FOR BEST RESULTS, ENTER SMALLEST NUMBER FIRST"
00050 INPUT A,B
00060 PRINT "ENTER PHASE - DEGREES"
00070 INPUT P
00080 PRINT "ENTER TOLERANCE - CONTROLS THE THICKNESS OF THE CURVE"
00090 PRINT "A GOOD VALUE IS BETWEEN 0 AND .2"
00100 INPUT T
00110 GO TO 370
00120 FOR Y=-32 TO 32
00130 F=ASN(-Y/32)
00140 H,K=0
00150 K=K+1
00160 FOR X=0 TO 1
00170 N=(X*(&PI-2*F)+F+2*&PI*K)*A/B+RAD(P)
00180 FOR C=-.5 TO .5
00190 FOR J=0 TO T*A/B STEP 3/80
00200 R=INT(SIN(N+J*C)*160/3+200/3)
00210 M(R)=1
00220 IF R<H THEN 240
00230 H=R
00240 NEXT J
00250 NEXT C
00260 NEXT X
00270 IF K*A/B>INT(K*A/B) THEN 150
00280 FOR X=1 TO H
00290 IF M(X)=1 THEN 320
00300 PRINT " ";
00310 GO TO 340
00320 PRINT "*";
00330 M(X)=0
00340 NEXT X
00350 PRINT
00360 NEXT Y
00370 FOR J=1 TO 5
00380 PRINT
00390 NEXT J
00400 IF Y=0 THEN 120
00410 END
```



ART & MATHEMATICAL STRUCTURES



FLIPS AND SPINS

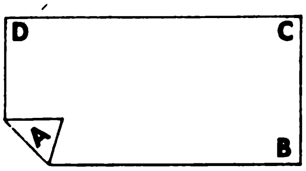
by R. Chandhok and M. Critchfield

Reprinted from Soloworks Module #2160 — PROJECT SOLO, Dept. of Computer Science, University of Pittsburgh, Pgh., PA 15260

Polygons: The Algebra of Symmetry

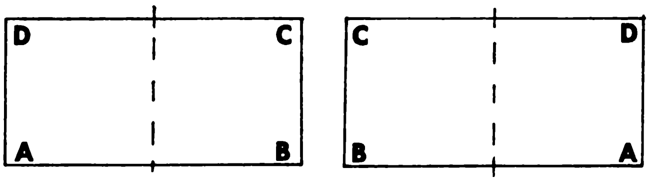
There is an important mathematical system that is not based on numbers, but on changing the position of a given polygon.

Imagine a rectangle that is lettered on both sides with the same letter in the same corner. In how many ways can you pick up the rectangle lettered ABCD and then "spin" it or "flip" it so that when you put it down the shape looks the same but the letters are in different positions.



Exercise: Make yourself a paper rectangle like the one above, use it to try out the manipulations discussed below.

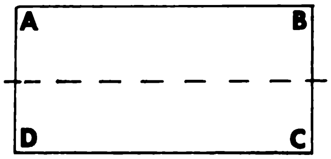
One way is to flip it on its vertical axis of symmetry:



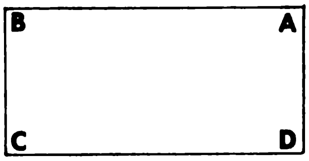
BEFORE FLIP AFTER FLIP

This is the "V" (flip on vertical axis) configuration. The other possible motions (and their resulting configurations) are:

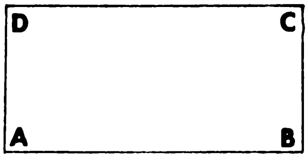
"H" (flip on horizontal axis)



"R" (rotate rectangle 180°)



"I" (itself, a rotation of 0° or no flip—the identity)



These are the elements of the system:

I, R, V, H

The binary operation of following one motion by another will allow us to create a table like this:

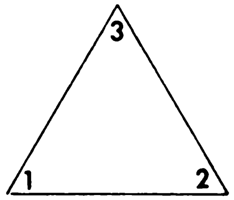
"followed by"	I	R	V	H
I	I		V	H
R	R			V
V		H	F	
H		V		

Exercise: Finish the table by manipulating your rectangle. What are the properties of this table?

Regular Polygons

From here on we will be considering only the regular polygons.

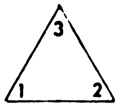
Exercise: Make a paper equilateral triangle. Instead of lettering the corners, number them. Can you do the same motions with it as with the rectangle?



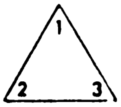
For the sake of simplicity, we will start using the following notations (which can be used for all regular polygons).

R_0 will be a rotation of 0 degrees (replaces I)
R_k will be a rotation of $k * (360/n)$ degrees
where n is the number of sides of the polygon

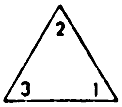
So, for $n = 3$, a triangle, there are three possible rotations or spins.



$R_0 = 0^\circ$ rotation $0 * (360/3) = 0$

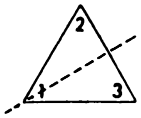


$R_1 = 120^\circ$ rotation $1 * (360/3) = 120$

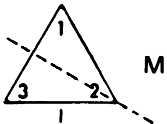


$R_2 = 240^\circ$ rotation $2 * (360/3) = 240$

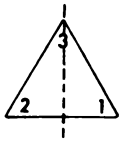
There are also three flips, and a new notation for them.



M_1



M_2 (M stands for "mirror" reflection since flips give the same result as a reflection)



M_3

A table for the 3-sided regular polygon (alias equilateral triangle) would look like this:

"followed by"	R_0	R_1	R_2	M_1	M_2	M_3
R_0						
R_1						
R_2						
M_1						
M_2						
M_3						

Exercise: You guessed it! Use your paper triangle to fill in the above table. Look for its properties.

You can see that it would get quite tedious to try to uncover the properties of all the regular polygons by hand this way.

However, try one more, since there is a slight variation when the number of sides is *even*.

Symmetries of the Square

Using the formula previously given, with n = 4 you can find that the number of rotations of the square are 4—0°, 90°, 180°, and 270°.

Exercise: Make a paper square. How many flips (reflections) does it have? Can you classify them into two kinds? Make a table like the one for the triangle with the spins and flips in the following order:

$$R_0, R_1, R_2, R_3, D_1, D_2, M_1, M_2$$

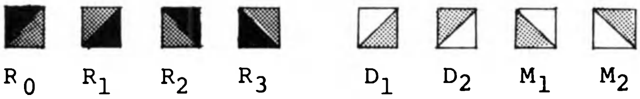
where D stands for "diagonal flip" and M, for a flip that bisects opposite sides.

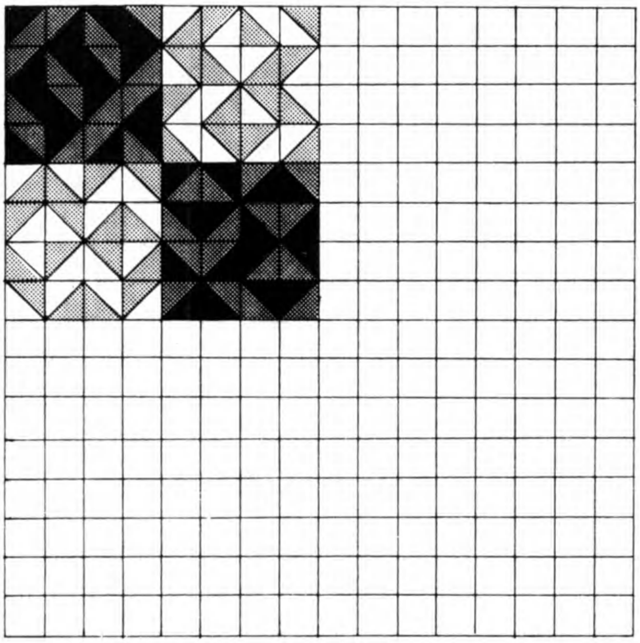
Note: It is possible to write a computer program to produce this, and all other tables from the symmetries of regular polygons. However, it is usually a good idea to go through the construction of tables, by hand, before trying to write a program. An annotated listing and a run of one such program is included at the end of this module.

Exercise: (optional) Make tables for the pentagon, hexagon, octagon, etc.

Designs from Symmetries

Designs based on these tables can be quite surprising and beautiful.





Exercise: (NOT OPTIONAL) Fill in the rest of the above design using colored markers. You decide whether to repeat, reflect, or rotate the original design.

Exercise: (optional) There are other plane figures which have symmetry but are not regular polygons, such as rectangles, parallelograms, trapezoids, and rhombuses. Devise a computer program to produce the table for one of these figures.

ENTER THE COMPUTER

Here's a RUN of the computer program we promised earlier. The program listing is on the following page.

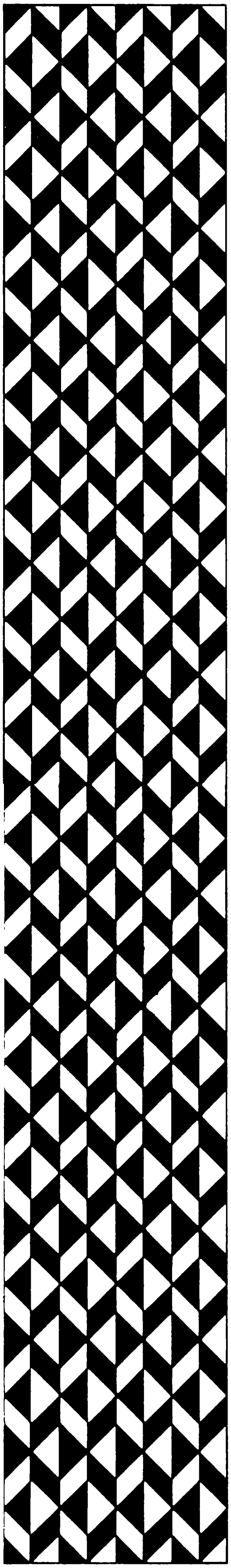
```
? 3

R0  = 1  2  3
R1  = 2  3  1
R2  = 3  1  2

M1  = 1  3  2
M2  = 3  2  1
M3  = 2  1  3

R0   R1   R2   M1   M2   M3

R1   R2   R0   M2   M3   M1

R2   R0   R1   M3   M1   M2

M1   M3   M2   R0   R2   R1

M2   M1   M3   R1   R0   R2

M3   M2   M1   R2   R1   R0

? 4

R0  = 1  2  3  4
R1  = 2  3  4  1
R2  = 3  4  1  2
R3  = 4  1  2  3

D1  = 1  4  3  2
D2  = 3  2  1  4
M1  = 2  1  4  3
M2  = 4  3  2  1

R0   R1   R2   R3   D1   D2   M1   M2

R1   R2   R3   R0   M2   M1   D1   D2

R2   R3   R0   R1   D2   D1   M2   M1

R3   R0   R1   R2   M1   M2   D2   D1

D1   M1   D2   M2   R0   R2   R1   R3

D2   M2   D1   M1   R2   R0   R3   R1

M1   D2   M2   D1   R3   R1   R0   R2

M2   D1   M1   D2   R1   R3   R2   R0
```

FURTHER READING:

A First Course in Abstract Algebra, by John S. Fraleigh, Addison-Wesley, 1967.

Mathematical Reasoning, Anita Harnadek, Midwest Publications, 1972.

```
LIST
POLY4   04:26 PM          16-DEC-75
10 PRINT"THIS PROGRAM CALCULATES THE ROTATIONS AND REFLECTIONS
11 PRINT"OF THE REGULAR POLYGONS .
12 PRINT "WHAT NUMBER OF SIDES (>2) DO YOU WANT?"
15 F$="!#  "
16 R$="R"
17 D$="D"
18 M$="M"
20 INPUT N
30 PRINT
40 IF 360/N=INT(360/N)THEN 80
50 PRINT "THERE IS NO REGULAR POLYGON OF";N;"SIDES WITH INTEGRAL ANGLES"
60 GO TO 20
70 DIM C(20,10)
80 D=0
85 REM CALCULATES THE ROTATIONS
90 FOR V=0 TO N-1
100 FOR I= 1 TO N
110 LET C(V+1,I)=I
120 NEXT I
130 FOR L= 1 TO N
140 Q=L+V
150 IF Q>N THEN Q=Q-N
160 C(V+1,L)=Q
170 NEXT L
180 PRINTUSING F$,R$,V;:PRINT"=";
190 FOR I= 1TO N
200 PRINT C(V+1,I);
210 NEXT I
220 PRINT
230 NEXT V
240 PRINT
245 REM CALCULATES THE FLIPS
250 IF N/2 = INT(N/2) THEN D=0 ELSE D=1
260 IF N02 = INT(N/2) THEN A=N/2 ELSE A=N
270 FOR I= 1 TO N
280 P= N+2-I
290 IF P>N THEN P=P-N
310 C(N+1,I)=P
320 NEXT I
330 V=1
335 K=N
340 GO SUB 450
350 FOR K=N+1 TO 2*N-1
360 FOR L= 1 TO N
370 C(K+1,L)= C(K,L)+2
380 IF K=3*N/2 THEN C(K+1,L)=C(K+1,L)+1
390 IF C(K+1,L)>N THEN C(K+1,L)=C(K+1,L)-N
400 NEXT L
410 V=V+1
420 GOSUB 450
430 NEXT K
440 GO TO 540
450 REM PRINTS OUT D'S AND M'S
460 IF V>A THEN D=D+1
470 IF V>A THEN V=1
480 IF D>1 THEN 540
490 IF D>0 THEN PRINTUSING F$,M$,V; :PRINT"=";:GO TO 510
500 PRINTUSING F$,D$,V;:PRINT"=";
510 FOR I=1 TO N: PRINT C(K+1,I);:NEXT I:PRINT
530 RETURN
540 PRINT
550 REM CALCULATES THE PRODUCT USING FIRST 2 NUMBERS
560 FOR K=1 TO 2*N
570 FOR K1= 1 TO 2*N
580 FOR I=1 TO 2
590 J=C(K,I)
600 T(I)=C(K1,J)
610 NEXT I
620 REM NEXT PART RECOGNIZES ANDPRINTS RESULT
630 IF T(1)<>N-1 THEN 650
640 IF T(2)=T(1)+1 THEN 720
650 IF T(2)=(T(1)+1)-INT(((T(1)+1)/N))*N THEN 720
660 IFT(1)/2=INT(T(1)/2) THEN T(1)= (T(1)+1+N)/2 ELSE T(1)=(T(1)+1)/2
670 IF N/2<>INT(N/2) THEN 700
680 IF T(1)<=N/2 THEN PRINTUSING F$,D$,T(1);:GO TO 730
690 PRINTUSING F$,M$,INT(T(1)-N/2);: GO TO 730
700 PRINTUSING F$,M$,T(1);
710 GO TO 730
720 PRINTUSING F$,R$LT(1)-1;
730 NEXT K1
740 PRINT:PRINT
750 NEXT K
760 GO TO 20
770 END
```



Musical Magic Squares

by

Fred T. Hofstetter

In music, the term "magic square" refers to the compositional matrix used by composers of 12-tone music. The twelve-tone school was started by Arnold Schoenberg, who discovered that if music was composed such that one of the twelve tones is repeated until every other tone is used, an atonal texture would result. Atonal means that there is no tonic, or that there is no "do re mi."

A twelve-tone row is a set of numbers from 1 through 12 typed in any order with no repetitions. Twelve-tone music uses a row and permutations of the row. The permutations consist of transpositions which are the pattern of the original row started on different notes; retrogrades which are the original row and the transpositions played backwards; inversions which are mirror-images of the original row and its transpositions; and retrograde inversions which are the inversions played backwards. The magic square program asks for a twelve-tone row. It then computes the transpositions, inversions, retrograde inversions, and retrogrades of the row, and prints them out as a matrix. The matrix is read as follows:

1. Reading rows from left to right yields transpositions.
2. Reading rows from right to left yields retrogrades.
3. Reading columns from top to bottom yields inversions.
4. Reading columns from botton to top yields retrograde inversions.

Two special symbols are used in the matrix: N means natural and ? means flat. It takes about half an hour to make a magic square by hand. The computer does it in just a second.

Program Listing

```
PRINT MASQUA

1000 REM    MAGIC SQUARE GENERATOR
1010 REM
1020 DEF FNM(X,Y)=X-Y*INT(X/Y)
1030 REM    MODULUS FUNCTION GIVES THE REMAINDER OF X DIVIDED BY Y
1040 DIM A(12)
1050 PRINT
1060 PRINT
1070 PRINT "WHAT ARE THE ROW NUMBERS";
1080 MAT INPUT A
1090 REM    INPUT THE MAIN ROW
1100 PRINT
1110 PRINT
1120 PRINT
1130 PRINT             M A G I C   S Q U A R E"
1140 PRINT
1150 PRINT
1160 LET B=1
1170 REM    B IS THE COUNTER FOR THE ROWS OF THE SQUARE
1180 LET C=0
1190 REM    C KEEPS TRACK OF THE DIFFERENCES BETWEEN SUCCESSIVE ROWS
1200 FOR D=1 TO 12
1210 REM    D IS THE COUNTER FOR INDIVIDUAL ELEMENTS OF THE ROWS
1220 LET E=FNM(C+A(D)-1,12)
1230 REM    E IS THE ACTUAL NOTE THAT SHOULD BE PRINTED
1240 IF E=0 THEN PRINT "CN  ";
1250 IF E=1 THEN PRINT "C#  ";
1260 IF E=2 THEN PRINT "DN  ";
1270 IF E=3 THEN PRINT "E?  ";
1280 IF E=4 THEN PRINT "EN  ";
1290 IF E=5 THEN PRINT "FN  ";
1300 IF E=6 THEN PRINT "F#  ";
1310 IF E=7 THEN PRINT "GN  ";
1320 IF E=8 THEN PRINT "A?  ";
1330 IF E=9 THEN PRINT "AN  ";
1340 IF E=10 THEN PRINT "B?  ";
1350 IF E=11 THEN PRINT "BN  ";
1360 NEXT D
1370 PRINT
1380 PRINT
1390 IF B>11 GOTO 1050
1400 REM    IF FINISHED, ASK FOR THE NEXT SQUARE. . .
1410 LET B=B+1
1420 LET C=C+A(B-1)-A(B)+12
1430 REM    OTHERWISE, GO BACK AND DO THE NEXT ROW
1440 GOTO 1200
#
```

N = Natural
? = Flat
# = Sharp

Sample Run

```
WHAT ARE THE ROW NUMBERS?  3,7,11,1,10,8,4,9,6,5,2,12


             M A G I C   S Q U A R E


DN  F#  B?  CN  AN  GN  E?  A?  FN  EN  C#  BN

B?  DN  F#  A?  FN  E?  BN  EN  C#  CN  AN  GN

F#  B?  DN  EN  C#  BN  GN  CN  AN  A?  FN  E?

EN  A?  CN  DN  BN  AN  FN  B?  GN  F#  E?  C#

GN  BN  E?  FN  DN  CN  A?  C#  B?  AN  F#  EN

AN  C#  FN  GN  EN  DN  B?  E?  CN  BN  A?  F#

C#  FN  AN  BN  A?  F#  DN  GN  EN  E?  CN  B?

A?  CN  EN  F#  E?  C#  AN  DN  BN  B?  GN  FN

BN  E?  GN  AN  F#  EN  CN  FN  DN  C#  B?  A?

CN  EN  A?  B?  GN  FN  C#  F#  E?  DN  BN  AN

E?  GN  BN  C#  B?  A?  EN  AN  F#  FN  DN  CN

FN  AN  C#  E?  CN  B?  F#  BN  A?  GN  EN  DN
```

Scales

All you've ever wanted to see (if its 11) and didn't have a program to ask for...

by
Marvin S. Thostenson

Language: HP 2000/Access BASIC

To practice spelling and observe the differences among the scales, use this program. It generates 11 types of scales: major, natural minor, harmonic minor, Hungarian minor, dorian, phygian, lydian, mixolydian, locrian, and whole tone.

When you run this program, you will be asked, "Which type of scale is wanted?" Respond by typing the first two letters of the name of the desired scale followed immediately by the desired key. Use a lower case (b' for the flat and use '#' for the sharp. Sample in puts would be *phe* for phrygian starting on E, *maf#* for major on F-sharp, and *whg* for whole tone on G.

The Author, Marvin S. Thostenson, is at the School of Music, University of Iowa, Iowa City, Iowa S2242

Sample Run

```
ELEVEN SCALE TYPES -- MAJOR, MINOR, MODAL, AND WHOLE TONE

This program prints in letter names, one octave upward, the major,
the natural, harmonic, melodic, and Hungarian minors, the
dorian, phrygian, lydian, mixolydian, and locrian modes, and
the whole tone scales.

Use a 3- or 4-character input; the first 2 char's are the scale
type, and the 3rd char'r is the single letter tonic, or the
last two char's are the tonic degree or the key signature.
SCALE TYPES-- ma na ha me do ph ly mi lo hu and wh
Input either a tonic or a signature.
EXAMPLES: macb lydb mieb whgb naf# hag# mea# loc# doc phd hue


WHICH TYPE OF SCALE IS WANTED?
?madb

   SCALE ASKED --------Major scale on Db

   ANSWER (in letter names) ----------


        Db  Eb  F   Gb  Ab  Bb  C   Db


WHICH TYPE OF SCALE IS WANTED?
?nae

   SCALE ASKED --------Nat'l minor scale on E

   ANSWER (in letter names) ----------


        E   F#  G   A   B   C   D   E


WHICH TYPE OF SCALE IS WANTED?
?ha2#

   SCALE ASKED --------Harm'c minor scale on B

   ANSWER (in letter names) ----------


        B   C#  D   E   F#  G   A#  B


WHICH TYPE OF SCALE IS WANTED?
?doeb

   SCALE ASKED --------Dorian mode on Eb

   ANSWER (in letter names) ----------


        Eb  F   Gb  Ab  Bb  C   Db  Eb
```

Program Listing

```
10  REM *** GENERATOR PROGRAM FOR ELEVEN TYPES OF SCALES ***
20  REM Written by Marvin S. Thostenson, Assoc. Prof., School
30  REM of Music, University of Iowa, Iowa City, Iowa 52242.
60  DIM A$[10],B$[50],C$[50],D$[65],E$[65]
70  DIM F$[72],G$[72],H$[72],I$[60],J$[50]
130  DIM K$[40],L$[40],M$[50],N$[40],O$[40]
133  DIM B1$[20],C1$[10],D1$[10],E1$[10],F1$[10]
134  DIM G1$[10],H1$[10],I1$[10],AO$[30],BO$[20],OO$[30]
140  A=B=C=D=E=H=K=L=M=N=O=P=Q=R=T=U=V=X=Y=Z=0
200  N=4
210  PRINT "ELEVEN SCALE TYPES -- MAJOR, MINOR, MODAL, AND WHOLE TONE";LIN(1)
220  PRINT "This program prints in letter names, one octave upward, the major,"
225  PRINT "the natural, harmonic, melodic, and Hungarian minors, the"
227  PRINT "dorian, phrygian, lydian, mixolydian, and locrian modes, and"
229  PRINT "the whole tone scales.";LIN(1)
230  PRINT "Use a 3- or 4-character input; the first 2 char's are the scale"
232  PRINT "type, and the 3rd char'r is the single letter tonic, or the"
234  PRINT "last two char's are the tonic degree or the key signature."
240  PRINT "SCALE TYPES-- ma na ha me do ph ly mi lo hu and wh"
250  PRINT "Input either a tonic or a signature."
260  PRINT "EXAMPLES: macb lydb mieb whgb naf# hag# mea# loc# doc phd hue"
270  B$="SCALE ASKED --------"
280  C$="ANSWER (in letter names) ----------"
290  O$[1,14]="STRUCTURE---- "
300  K$=" tetrachords"
310  U=1
320  PRINT TAB(8);LIN(2),"WHICH TYPE OF SCALE IS WANTED?"
330  INPUT A$
340  N=LEN(A$)
350  IF A$="stop" THEN 1290
360  E$="manahamedophlymilohuwh"
370  FOR X=1 TO 22 STEP 2
380  IF A$[1,2]=E$[X,X+1] THEN 400
390  NEXT X
400  Q=(X+1)/2
410  AO$=A$
420  AO$[1,1]=UPS$(AO$[1,1])
430  READ D$
440  AO$=A$
450  AO$[1,1]=UPS$(AO$[1,1])
460  IF D$[1,2]=AO$[1,2] THEN 480
470  GOTO 430
480  J$=D$
490  RESTORE
500  IF N=3 THEN 520
510  GOTO 530
520  A$[4,4]=" "
530  IF Q=1 OR Q=7 OR Q=8 OR Q=11 THEN 550
540  IF Q >= 2 AND Q <= 6 OR Q=9 OR Q=10 THEN 570
550  Y=1
560  GOTO 580
570  Y=2
580  F$="oxexaxdxgxcxfxb#e#a#d#g#c#f#b e a d g c f bbebabdbgbcbfbbdedadddgdcd"
590  G$="BxExAxDxGxCxFxB#E#A#D#G#C#F#B E A D G C F BbEbAbDbGbCbFbBdEdAdDdGdCd"
600  GOTO Y OF 610,630
610  H$="5t4t3t2t1t7x6x5x4x3x2x1x7#6#5#4#3#2#1#0#1b2b3b4b5b6b7b1d2d3d4d5d6d7d"
620  GOTO 640
630  H$="2t1t7x6x5x4x3x2x1x7#6#5#4#3#2#1#0#1b2b3b4b5b6b7b1d2d3d4d5d6d7d8d9d  "
640  FOR V=1 TO 68 STEP 2
650  IF A$[3,4]=F$[V,V+1] THEN 680
660  IF A$[3,4]=H$[V,V+1] THEN 680
670  NEXT V
680  C1$=G$[V,V+1]
690  T=T+1
700  IF T=9 THEN 1160
710  GOTO T OF 720,740,790,840,890,940,990,1040
720  R=0
730  GOTO 1060
740  IF Q=6 OR Q=9 THEN 770
750  R=-4
760  GOTO 1060
770  R=10
780  GOTO 1060
790  IF Q=1 OR Q=7 OR Q=8 OR Q=11 THEN 820
800  R=6
810  GOTO 1060
820  R=-8
830  GOTO 1060
840  IF Q=7 OR Q=10 OR Q=11 THEN 870
850  R=2
860  GOTO 1060
870  R=-12
880  GOTO 1060
890  IF Q=9 'OR Q=11 THEN 920
900  R=-2
910  GOTO 1060
920  R=12
930  GOTO 1060
940  IF Q=1 OR Q=4 OR Q=5 OR Q=7 OR Q=8 THEN 970
950  R=8
960  GOTO 1060
970  R=-6
980  GOTO 1060
990  IF Q=1 OR Q=3 OR Q=4 OR Q=7 OR Q=10 THEN 1020
1000  R=4
1010  GOTO 1060
1020  R=-10
1030  GOTO 1060
1040  R=0
1050  GOTO 1060
1060  IF Q=11 AND T=5 THEN 1090
1070  I$[U,U+1]=G$[V+R,V+R+1]
1080  GOTO 1100
1090  GOTO 690
1100  I$[U+2,U+3]="  "
1110  IF I$[U+1,U+1]="d" THEN 1130
1120  GOTO 1140
1130  I$[U+1,U+2]="bb"
1140  U=U+4
1150  GOTO 690
1160  PRINT LIN(1),TAB(3),B$;J$;C1$;LIN(1)
1170  PRINT TAB(3);C$;LIN(1)
1180  PRINT LIN(1);TAB(8);I$
1190  C1$=""
1200  I$=""
1210  G$=""
1220  Q=T=R=0
1230  PRINT
1240  GOTO 310
1250  DATA "Major scale on ","Nat'l minor scale on ","Harm'c minor scale on "
1260  DATA "Mel'c minor scale on ","Dorian mode on ","Phrygian mode on "
1270  DATA "Lydian mode on ","Mixolydian mode on ","Locrian mode on "
1280  DATA "Hung'n minor scale on ","Whole tone scale on "
1290  END
```

Psychoanlysis (?) by Computer...

ELIZA

Steve North

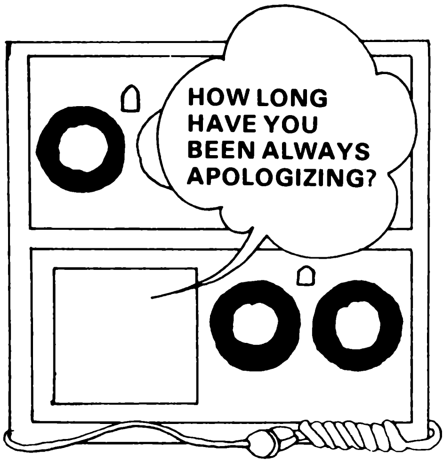
Sample Run

```
RUN
HI!  I'M ELIZA.  WHAT'S YOUR PROBLEM?
? I AM AN IBM 370
DID YOU COME TO ME BECAUSE YOU ARE AN IBM 370
? YES I WANT A NEW LINEPRINTER
WHAT WOULD IT MEAN TO YOU IF YOU GOT A NEW LINEPRINTFR
? ABOUT 1200 LPM
SAY, DO YOU HAVE ANY PSYCHOLOGICAL PROBLEMS?
? WHAT WOULD YOU KNOW I'M A 370 AND YOU'RE AN IMSAI
WHAT MAKES YOU THINK I AM AN IMSAI
? IT SAYS THAT ON YOUR FRONT PANEL
WHY ARE YOU CONCERNED ABOUT MY FRONT PANEL
? LOOK YOU BROUGHT UP THE SUBJECT
WE WERE DISCUSSING YOU-- NOT ME.
? I'M FIXING A HOLE...
DID YOU COME TO ME BECAUSE YOU ARE FIXING A HOLE...
? SORRY THAT WAS A TEN YEAR OLD SOFTWARE BUG
PLEASE DON'T APOLOGIZE!
? I AM ALWAYS APOLOGIZING
HOW LONG HAVE YOU BEEN ALWAYS APOLOGIZING
? SINCE MY FIRST COLD START
WHAT DOES THAT SUGGEST TO YOU?
? YOU'RE THE SHRINK
DOES IT PLEASE YOU TO BELIEVE I AM THE SHRINK
? DONT YOU KNOW?
OH, I KNOW?
? I AM BEGINNING TO WONDER ABOUT YOU
DO YOU BELIEVE IT IS NORMAL TO BE BEGINNING TO WONDER ABOUT I
? HA HA HA YOU CAN'T EVEN TALK PROPERLY
YOU'RE NOT REALLY TALKING ABOUT ME, ARE YOU?
? IS THERE ANYONE ELSE IN THE ROOM?
I SEE.
? COMPUTERS DONT HAVE EYES
ARE YOU SURE?
? ABOUT COMPUTERS?
DO COMPUTERS WORRY YOU?
? SHOULD THEY AFTER ALL I AM A 370
DO YOU ENJOY BEING A 370
?
```

Language: BASIC (MITS 8K)

Author: Originally programmed in LISP by Joseph Weizenbaum. This version is based on one written by Jeff Shrager.

Description: ELIZA is a program that accepts natural English as input and carries on a reasonably coherent conversation based on the psychoanalytic techniques of Carl Rogers. You will have to forgive ELIZA for being a poor English student. You'll find that it is best not to use punctuation in your input, and you'll have to carry the conversation. But it does work!


HOW LONG HAVE YOU BEEN ALWAYS APOLOGIZING?

Remarks: In order to do what it does, ELIZA must: (1) get a string from the user, and prepare it for further processing: (2) find the keywords in the input string: (3) if a keyword is found, take the part of the string following the keyword and "translate" all the personal pronouns and verbs ("I" becomes "YOU", "ARE" becomes "AM", etc.); (4) finally, look up an appropriate reply based on the keyword which was found, print it and, if necessary, the "translated" string. ELIZA uses four types of program data to accomplish this:

(1) 36 keyword, such as "I AM", "WHY DONT YOU", and "COMPUTER". The keywords must be in order of priority, so ELIZA will key on "YOU ARE" before "YOU".

(2) 12 strings used for the translation or conjugation process. These are in pairs

Program Listing

```
10 REM
20 REM      ELIZA/DOCTOR
30 REM      CREATED BY JOSEPH WEIZENBAUM
40 REM      THIS VERSION BY JEFF SHRAGER
50 REM      EDITIED AND MODIFIED FOR MITS 8K BASIC 4.0 BY STEVE NORTH
60 REM      CREATIVE COMPUTING PO BOX 789-M MORRISTOWN NJ 07960
70 REM
80 REM      -----INITIALIZATION-----
90 DIM C$(72),I$(72),K$(72),F$(72),S$(72),R$(72),P$(72),Z$(72)
100 DIM S(36),R(36),N(36)
110 N1=36:N2=12:N3=112
120 FOR X=1 TO N1+N2+N3:READ Z$:NEXT X:REM SAME AS RESTORE
130 FORX=1 TO N1
140 READ S(X),L:R(X)=S(X):N(X)=S(X)+L-1
150 NEXT X
160 PRINT "HI!  I'M ELIZA.  WHAT'S YOUR PROBLEM?"
170 REM
180 REM      -----USER INPUT SECTION-----
190 REM
200 INPUT I$
201 I$=" "+I$+"  "
210 REM  GET RID OF APOSTROPHES
220 FOR L=1 TO LEN(I$)
230 IFMID$(I$,L,1)="'"THENI$=LEFT$(I$,L-1)+RIGHT$(I$,LEN(I$)-L):GOTO230
240 IFL+4<=LEN(I$)THENIFMID$(I$,L,4)="SHUT"THENPRINT"SHUT UP...":END
```

such that if one member of the pair is found, the other is substituted for it. Examples: "Y", "YOU", "AM", "ARE", etc.

(3) 112 reply strings. The strings are arranged in groups corresponding to the keywords. There is no fixed number of different replies for each keyword. Replies ending in a "*" are to be followed by the translated string, while the strings ending in normal punctuation are to be printed alone.

(4) Numerical data to determine which replies to print for each keyword. For each keyword there is a pair of numbers signifying (start of reply strings, number of reply strings). Thus the fifth pair of number, (10,4), means that the replies for the fifth keyword ("I DONT") start with the tenth reply string, and that there are four replies.

Detailed Explanation:

Lines 10-160: Initialization. Arrays and strings are dimensioned. N1, N2, and N3, which represent the number of keywords, number of translation strings, and number of replies respectively, are defined. Then the arrays are filled. S(keyword number) is the ordinal number of the start of the reply strings for a given keyword, R(keyword number) is the actual reply to be used next, and N(keyword number) is the last reply for that keyword. Finally an introduction is printed.

Lines 170-255: User input section. This part of the program gets a string from the user, places a space at the start of the string and two at the end (to make it easier to correctly locate keywords and to preventsubscripting out of bounds), throws out all the apostrophes (so DONT and DON'T are equivalent), and stops if the word SHUT is found in the input string (which it takes to mean SHUT UP). ELIZA also checks for repetitive input by the user.

Lines 260-370: Keyword-finding section. ELIZA scans the input string for keywords and saves the keyword of highest priority temporarily in S, T, and F$. If no keyword is found, the keyword defaults to number 36, NOKEYFOUND (which causes ELIZA to say something noncommital) and it skips the next section.

Lines 380-555: Translation or Conjugation section. The part of the input string following the keyword is saved. Then pairs of translation strings, as described above, are read and upon the occurence of one of these strings, the other is substituted for it. When this is done ELIZA makes sure there is only one leading space in the translated string.

Lines 560-640: Reply printing section. Using R(keyword number), S(keyword number), and N(keyword number), the correct reply is located. The pointer for the next reply is bumped and reset if it is

```
250 NEXT L
255 IF I$=P$ THEN PRINT "PLEASE DON'T REPEAT YOURSELF!":GOTO 170
260 REM
270 REM     -----FIND KEYWORD IN I$-----
280 REM
290 RESTORE
295 S=0
300 FOR K=1 TO N1
310 READ K$
315 IF S>0 THEN360
320 FOR L=1 TO LEN(I$)-LEN(K$)+1
340 IF MID$(I$,L,LEN(K$))=K$THENS=K:T=L:F$=K$
350 NEXT L
360 NEXT K
365 IF S>0 THEN K=S:L=T:GOTO390
370 K=36:GOTO570:REM  WE DIDN'T FIND ANY KEYWORDS
380 REM
390 REM     TAKE RIGHT PART OF STRING AND CONJUGATE IT
400 REM     USING THE LIST OF STRINGS TO BE SWAPPED
410 REM
420 RESTORE:FORX=1 TO N1:READ Z$:NEXT X:REM SKIP OVER KEYWORDS
430 C$=" "+RIGHT$(I$,LEN(I$)-LEN(F$)-L+1)
440 FOR X=1 TO N2/2
450 READ S$,R$
460 FOR L= 1 TO LEN(C$)
470 IF L+LEN(S$)>LEN(C$) THEN 510
480 IF MID$(C$,L,LEN(S$))<>S$ THEN 510
490 C$=LEFT$(C$,L-1)+R$+RIGHT$(C$,LEN(C$)-L-LEN(S$)+1)
495 L=L+LEN(R$)
500 GOTO 540
510 IF L+LEN(R$)>LEN(C$)THEN540
520 IF MID$(C$,L,LEN(R$))<>R$ THEN 540
530 C$=LEFT$(C$,L-1)+S$+RIGHT$(C$,LEN(C$)-L-LEN(R$)+1)
535 L=L+LEN(S$)
540 NEXT L
550 NEXT X
555 IF MID$(C$,2,1)=" "THENC$=RIGHT$(C$,LEN(C$)-1):REM ONLY 1 SPACE
560 REM
570 REM     NOW USING THE KEYWORD NUMBER (K) GET REPLY
580 REM
590 RESTORE:FOR X= 1 TO N1+N2:READ Z$:NEXT X
600 FORX=1TOR(K):READ F$:NEXT X:REM  READ RIGHT REPLY
610 R(K)=R(K)+1: IFR(K)>N(K) THEN R(K)=S(K)
620 IF RIGHT$(F$,1)<>"*" THEN PRINT F$:P$=I$:GOTO 170
630 PRINT LEFT$(F$,LEN(F$)-1);C$
640 P$=I$:GOTO 170
1000 REM
1010 REM     -----PROGRAM DATA FOLLOWS-----
1020 REM
1030 REM     KEYWORDS
1040 REM
1050 DATA "CAN YOU","CAN I","YOU ARE","YOURE","I DONT","I FEEL"
1060 DATA "WHY DONT YOU","WHY CANT I","ARE YOU","I CANT","I AM","IM "
1070 DATA "YOU ","I WANT","WHAT","HOW","WHO","WHERE","WHEN","WHY"
1080 DATA "NAME","CAUSE","SORRY","DREAM","HELLO","HI ","MAYBE"
1090 DATA " NO","YOUR","ALWAYS","THINK","ALIKE","YES","FRIEND"
1100 DATA "COMPUTER","NOKEYFOUND"
1200 REM
1210 REM     STRING DATA FOR CONJUGATIONS
1220 REM
1230 DATA " ARE "," AM ","WERE ","WAS "," YOU "," I ","YOUR ","MY "
1235 DATA " IVE "," YOUVE "," IM "," YOURE "
1300 REM
1310 REM     REPLIES
1320 REM
1330 DATA "DON'T YOU BELIEVE THAT I CAN*"
1340 DATA "PERHAPS YOU WOULD LIKE TO BE ABLE TO*"
1350 DATA "YOU WANT ME TO BE ABLE TO*"
1360 DATA "PERHAPS YOU DON'T WANT TO*"
1365 DATA "DO YOU WANT TO BE ABLE TO*"
1370 DATA "WHAT MAKES YOU THINK I AM*"
1380 DATA "DOES IT PLEASE YOU TO BELIEVE I AM*"
1390 DATA "PERHAPS YOU WOULD LIKE TO BE*"
1400 DATA "DO YOU SOMETIMES WISH YOU WERE*"
1410 DATA "DON'T YOU REALLY*"
1420 DATA "WHY DON'T YOU*"
1430 DATA "DO YOU WISH TO BE ABLE TO*"
1440 DATA "DOES THAT TROUBLE YOU?"
1450 DATA "TELL ME MORE ABOUT SUCH FEELINGS."
1460 DATA "DO YOU OFTEN FEEL*"
1470 DATA "DO YOU ENJOY FEELING*"
1480 DATA "DO YOU REALLY BELIEVE I DON'T*"
1490 DATA "PERHAPS IN GOOD TIME I WILL*"
1500 DATA "DO YOU WANT ME TO*"
1510 DATA "DO YOU THINK YOU SHOULD BE ABLE TO*"
1520 DATA "WHY CAN'T YOU*"
1530 DATA "WHY ARE YOU INTERESTED IN WHETHER OR NOT I AM*"
1540 DATA "WOULD YOU PREFER IF I WERE NOT*"
1550 DATA "PERHAPS IN YOUR FANTASIES I AM*"
```

too large. If the reply string ends in a "*" it is printed with the translated string, otherwise it is printed alone. The previously entered input string is saved to permit checking for repetetive input, and then ELIZA goes back for more input.

Limitations: Runs in 16K of memory.

Modifications: You can easily add, change, or delete any of the keywords, translation words, or replies. Remember, you will also have to change N1, N2, N3, and/or the numerical data. Just as a suggestion, if you decide to insert "ME" and "YOU" in the translation string list, put a nonprinting (control) character in YOU to prevent ELIZA from substituting I→YOU→ME. This means that YOU will always be assumed to be the subject of a verb, never the object, but resolving that difficulty is a whole different problem.

A Few Comments: The structures found in lines 120, 420, and 590 could be replaced by RESTORE NNNN statements if your BASIC has them. The use of an INSTR, SEARCH, or POS function to determine if one string is a substring of another would probably speed things up considerably (it takes ELIZA around 10 seconds to think of a reply).

What it all means: we'll leave to you. Although this program is an inferior imitation of the original, it does work. It is pretty farfetched to believe that a psycholanalyst is nothing but a sentence-input-keyword-finder-conjugator-reply finder, but if you really think so, you can buy your computer a speech-recognition unit, a Computalker and a green couch, and charge $75/hr! My computer, the doctor!

MITS STRING BASIC FUNCTIONS

LEFT$ (X$,X) takes the X leftmost characters of X$.

RIGHT$(X$,X) takes the X rightmost characters of X$.

MID$ (X$,X,Y) takes Y characters from X$ starting with the Xth character.

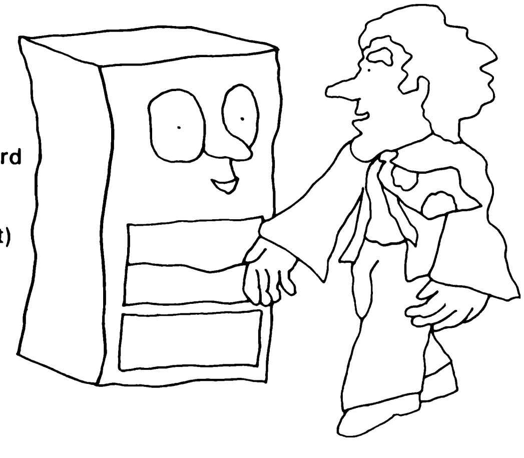
DIM X$ (72) dimensions X$ to hold 72 characters. That may seem obvious, but there's a story that goes along with this. The listing of ELIZA was made on a friend's SWTPC 6800 running 8K BASIC. After making a small patch in BASIC, loading a paper tape, and changing all the multiple statements to separate lines, we were all ready to try out ELIZA when Bob (whose computer it was asked about the enormous strings ELIZA used. It turns out that in SWTPC BASIC, DIM X$(72) means dimension string vector X$ to have 72 little strings (X$(K)) of 18 character

```
1560 DATA "HOW DO YOU KNOW YOU CAN'T*"
1570 DATA "HAVE YOU TRIED?"
1580 DATA "PERHAPS YOU CAN NOW*"
1590 DATA "DID YOU COME TO ME BECAUSE YOU ARE*"
1600 DATA "HOW LONG HAVE YOU BEEN*"
1610 DATA "DO YOU BELIEVE IT IS NORMAL TO BE*"
1620 DATA "DO YOU ENJOY BEING*"
1630 DATA "WE WERE DISCUSSING YOU-- NOT ME."
1640 DATA "OH, I*"
1650 DATA "YOU'RE NOT REALLY TALKING ABOUT ME, ARE YOU?"
1660 DATA "WHAT WOULD IT MEAN TO YOU IF YOU GOT*"
1670 DATA "WHY DO YOU WANT*"
1680 DATA "SUPPOSE YOU SOON GOT*"
1690 DATA "WHAT IF YOU NEVER GOT*"
1700 DATA "I SOMETIMES ALSO WANT*"
1710 DATA "WHY DO YOU ASK?"
1720 DATA "DOES THAT QUESTION INTEREST YOU?"
1730 DATA "WHAT ANSWER WOULD PLEASE YOU THE MOST?"
1740 DATA "WHAT DO YOU THINK?"
1750 DATA "ARE SUCH QUESTIONS ON YOUR MIND OFTEN?"
1760 DATA "WHAT IS IT THAT YOU REALLY WANT TO KNOW?"
1770 DATA "HAVE YOU ASKED ANYONE ELSE?"
1780 DATA "HAVE YOU ASKED SUCH QUESTIONS BEFORE?"
1790 DATA "WHAT ELSE COMES TO MIND WHEN YOU ASK THAT?"
1800 DATA "NAMES DON'T INTEREST ME."
1810 DATA "I DON'T CARE ABOUT NAMES-- PLEASE GO ON."
1820 DATA "IS THAT THE REAL REASON?"
1830 DATA "DON'T ANY OTHER REASONS COME TO MIND?"
1840 DATA "DOES THAT REASON EXPLAIN ANYTHING ELSE?"
1850 DATA "WHAT OTHER REASONS MIGHT THERE BE?"
1860 DATA "PLEASE DON'T APOLOGIZE!"
1870 DATA "APOLOGIES ARE NOT NECESSARY."
1880 DATA "WHAT FEELINGS DO YOU HAVE WHEN YOU APOLOGIZE."
1890 DATA "DON'T BE SO DEFENSIVE!"
1900 DATA "WHAT DOES THAT DREAM SUGGEST TO YOU?"
1910 DATA "DO YOU DREAM OFTEN?"
1920 DATA "WHAT PERSONS APPEAR IN YOUR DREAMS?"
1930 DATA "ARE YOU DISTURBED BY YOUR DREAMS?"
1940 DATA "HOW DO YOU DO ... PLEASE STATE YOUR PROBLEM."
1950 DATA "YOU DON'T SEEM QUITE CERTAIN."
1960 DATA "WHY THE UNCERTAIN TONE?"
1970 DATA "CAN'T YOU BE MORE POSITIVE?"
1980 DATA "YOU AREN'T SURE?"
1990 DATA "DON'T YOU KNOW?"
2000 DATA "ARE YOU SAYING NO JUST TO BE NEGATIVE?"
2010 DATA "YOU ARE BEING A BIT NEGATIVE."
2020 DATA "WHY NOT?"
2030 DATA "ARE YOU SURE?"
2040 DATA "WHY NO?"
2050 DATA "WHY ARE YOU CONCERNED ABOUT MY*"
2060 DATA "WHAT ABOUT YOUR OWN*"
2070 DATA "CAN YOU THINK OF A SPECIFIC EXAMPLE?"
2080 DATA "WHEN?"
2090 DATA "WHAT ARE YOU THINKING OF?"
2100 DATA "REALLY, ALWAYS?"
2110 DATA "DO YOU REALLY THINK SO?"
2120 DATA "BUT YOU ARE NOT SURE YOU*"
2130 DATA "DO YOU DOUBT YOU*"
2140 DATA "IN WHAT WAY?"
2150 DATA "WHAT RESEMBLANCE DO YOU SEE?"
2160 DATA "WHAT DOES THE SIMILARITY SUGGEST TO YOU?"
2170 DATA "WHAT OTHER CONNECTIONS DO YOU SEE?"
2180 DATA "COULD THERE REALLY BE SOME CONNECTION?"
2190 DATA "HOW?"
2200 DATA "YOU SEEM QUITE POSITIVE."
2210 DATA "ARE YOU SURE?"
2220 DATA "I SEE."
2230 DATA "I UNDERSTAND."
2240 DATA "WHY DO YOU BRING UP THE TOPIC OF FRIENDS?"
2250 DATA "DO YOUR FRIENDS WORRY YOU?"
2260 DATA "DO YOUR FRIENDS PICK ON YOU?"
2270 DATA "ARE YOU SURE YOU HAVE ANY FRIENDS?"
2280 DATA "DO YOU IMPOSE ON YOUR FRIENDS?"
2290 DATA "PERHAPS YOUR LOVE FOR FRIENDS WORRIES YOU."
2300 DATA "DO COMPUTERS WORRY YOU?"
2310 DATA "ARE YOU TALKING ABOUT ME IN PARTICULAR?"
2320 DATA "ARE YOU FRIGHTENED BY MACHINES?"
2330 DATA "WHY DO YOU MENTION COMPUTERS?"
2340 DATA "WHAT DO YOU THINK MACHINES HAVE TO DO WITH YOUR PROBLEM?"
2350 DATA "DON'T YOU THINK COMPUTERS CAN HELP PEOPLE?"
2360 DATA "WHAT IS IT ABOUT MACHINES THAT WORRIES YOU?"
2370 DATA "SAY, DO YOU HAVE ANY PSYCHOLOGICAL PROBLEMS?"
2380 DATA "WHAT DOES THAT SUGGEST TO YOU?"
2390 DATA "I SEE."
2400 DATA "I'M NOT SURE I UNDERSTAND YOU FULLY."
2410 DATA "COME COME ELUCIDATE YOUR THOUGHTS."
2420 DATA "CAN YOU ELABORATE ON THAT?"
2430 DATA "THAT IS QUITE INTERESTING."
2500 REM
```

each. There's no easy way to handle more than 18 characters at a time! Those of you using SWTPC 6800 BASIC should have a lot of fun with this.

```
2510 REM      DATA FOR FINDING RIGHT REPLIES
2520 REM
2530 DATA 1,3,4,2,6,4,6,4,10,4,14,3,17,3,20,2,22,3,25,3
2540 DATA 28,4,28,4,32,3,35,5,40,9,40,9,40,9,40,9,40,9,40,9
2550 DATA 49,2,51,4,55,4,59,4,63,1,63,1,64,5,69,5,74,2,76,4
2560 DATA 80,3,83,7,90,3,93,6,99,7,106,6
```

Name	*Usage*
R(X),S(X),N(X)	See Text
I$	Input string
K$	Keyword string
C$	Translated or conjugated string
F$	Reply string, also used to save K$ in scanning for keyword
R$,S$	Strings used in conjugation process
P$	Previous input string
Z$	Scratch (used for simulating RESTORE NNNN statement)
N1	Number of keywords
N2	Number of conjugation strings
N3	Number of replies
K	Keyword number
S,T	Used to save K and L when scanning for keyword
X,L	X,L Scratch. X is generally used for looping while L is used for scanning through strings
V	V Used for scanning for keyword string



A for Effort, Zero for Arab

Many years ago a Roman civil engineer, who was a high official in Alexandria, was approached by a young Arabian mathematician with an idea which the Easterner believed would be of much value to the Roman Government in its road-building, navigating, tax-collecting, and census-taking activities. As the Arab explained in his manuscript, he had discovered a new type of notation for number writing, which was inspired from some Hindu inscriptions.

The Roman official presumably studied this manuscript very carefully for several hours, then wrote his reply:

Your courier brought your proposal at a time when my duties were light, so fortunately I have had the opportunity to study it carefully, and am glad to be able to submit these detailed comments.

Your new notation may have a number of merits, as you claim, but it is doubtful whether it ever would be of any practical value to the Roman Empire. Even if authorized by the Emperor himself, as a proposal of this magnitude would have to be, it would be vigorously opposed by the populace, principally because those who had to use it would not sympathize with your radical ideas. Our scribes complain loudly that they have too many letters in the Roman Alphabet as it is, and now you propose these ten additional symbols of your number system, namely
1, 2, 3, 4, 5, 6, 7, 8, 9, and your 0.

It is clear that your 1-mark has the same meaning as our mark-I but since this mark-I already is a well-established character why is there any need for yours?

Then you explain that last circle-mark, like our letter O, as representing 'an empty column', or meaning nothing. If it means nothing, what is the purpose of writing it? I cannot see that it is serving any useful purpose; but to make sure, I asked my assistant to read this section, and he drew the same conclusion.

You say the number 01 means the same as just 1. This is an intolerable ambiguity and could not be permitted in any Roman legal documents. Your notation has other ambiguities which seem even worse: You explain that the mark-1 means ONE, yet on the very same page you show it to mean TEN in 10, and one HUNDRED in your 100. If my official duties had not been light while reading this, I would have stopped here; you must realize that examiners will not pay much attention to material containing such obvious errors.

Further on, you claim that your system enumeration is much simpler than the Roman Numerals. I regret to advise that I have examined this point very carefully and must conclude otherwise. For example, counting up to FIVE, you require five new symbols whereas we Romans accomplish this with just two old ones, the mark-I and the mark-V. At first sight the combination IV (meaning ONE before FIVE) for four may seem less direct than the old IIII, but note that this alert representation involves LESS EFFORT, and that gain is the conquering principle of the Empire.

Counting up to twenty (the commonest counting range among the populace), you require ten symbols whereas we now need but three; the I, V, and X. Note particularly the pictorial suggestiveness of the V as half of the X. Moreover, it is pictorially evident that XX means ten-and-ten, and this seems much preferred over your 20. These pictorial associations are very important to the lower classes, for as the African says, 'Picture tells thousand words'.

You claim that your numbers as a whole are briefer than the Roman Numerals, but this is not made evident in your proofs. Even if true, it is doubtful that this would mean much to the welfare of the Empire, since numbers account for only a small fraction of the written records; and in any case, there are plenty of slaves with plenty of time to do this work.

When you attempt to show that you can manipulate these numbers much more readily than Roman Numerals, your explanations are particularly bad and obscure. For example, you show in one addition that 2 and 3 equals 5, yet in the case you write as

$$\begin{array}{r}79\\+16\\\hline 95\end{array}$$

this indicates that 9 and 6 also equals 5. How can this be? While it is not clear, it is evident that the other part is in error, for 7 and 1 equal 8, not 9.

Your so-called 'repeating and dividing' tables also require much more explanation, and possibly correction of errors. I can see that your 'Nine Times' Table gives sets which add up to nine, namely 18, 27, 36, 45, 54, 63, 72, 81 and 90 but I see no such useful correlation in the 'Seven Times' Table, for example. Since we have SEVEN, not nine, days in the Roman Week, it seems far more important we have a system that gives more sensible combinations in this table.

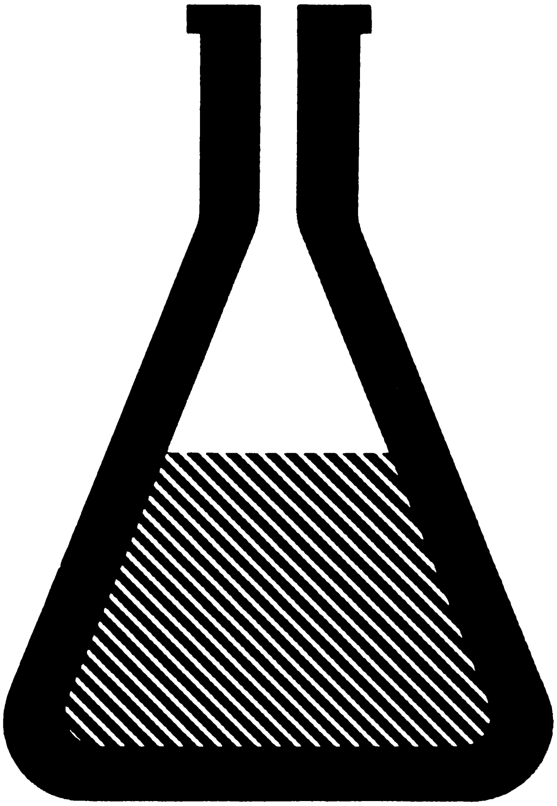
All in all, I would advise you to forget this overly ambitious proposal, return to your sand piles, and leave the number reckoning to the official Census Takers and Tax Collectors. I am sure that they give these matters a great deal more thought than you or I can.

William J. Wiswesser

A student shows how he used a Sol computer to assist other students in their lab work.

Inorganic Chemistry Program

J.P. Peer



Partial Sample Run

```
YOU CAN PICK THE DISPLAY SPEED FOR THIS PROGRAM.
'0' IS STANDARD (FAST DISPLAY), '99' IS VERY SLOW.
Type the display speed ( 0 to 99 ) here:  0


Type the number of Stability Constants ( 1 to 10 ) here:  5


   OKAY.  NOW TYPE IN THE STABILITY CONSTANTS.
Type K( 1 ) here:  1.9E 4
Type K( 2 ) here:  3.5E 3
Type K( 3 ) here:  7.9E 2
Type K( 4 ) here:  1.5E 2
Type K( 5 ) here:  0.28

NOW YOU NEED TO TYPE IN THE ANALYTICAL CONCENTRATIONS
OF THE LIGAND AND METAL ( PLEASE USE MOLES / LITER ).
Type the Ligand concentration ( 0 to 50 ) here:  1.0
Type the Metal concentration ( 0 to 50 ) here:  0.1


ALPHAS                      CONCENTRATIONS OF COMPLEXES
++++++                      ++++++++++++++++++++++++++
# 0 = 9.8625933 E-14  0-Ligand complex = 9.8625933 E-15 molar
# 1 = 1.8738927 E-9   1-Ligand complex = 1.8738927 E-10 molar
# 2 = .00000656       2-Ligand complex = .00000066 molar
# 3 = .00518131       3-Ligand complex = .00051813 molar
# 4 = .77719701       4-Ligand complex = .0777197 molar
# 5 = .21761516       5-Ligand complex = .02176152 molar
```

Program Listing

```
1000 REM CHM342C MODIFIED FOR SOL.  1977 BY J. P. PEER
1010 REM 417 WEST WATER STREET, BERNE, INDIANA  46711
1020 REM LET'S CLEAR THE CRT SCREEN.  TO DO THIS WE PRINT THE
1030 REM SCREEN-ERASE CHARACTER.  ON SOL IT IS THE 'CLEAR'
1040 REM KEY.  ON OTHER COMPUTERS IT IS THE 'SCREEN ERASE'
1050 REM KEY OR 'CONTROL-L.'  IF YOU ARE USING A HARD-COPY
1060 REM TERMINAL YOU CAN OMIT THIS.
1100 PRINT"(clear key)"
1110 REM SET SPEED OF SOL'S VIDEO DISPLAY MODULE (VDM).  IF
1120 REM YOUR MACHINE IS NOT A SOL YOU CAN OMIT THIS.
1200 SET S=0
1250 DIM K(11),A1(12)
1290 REM A COLON  (:)  SEPARATES STATEMENTS ON ONE LINE.
1300 PRINT:PRINT:PRINT
1400 PRINT"    H I !   T H I S   P R O G R A M   F I N D S   T H E"
1500 PRINT"C O N C E N T R A T I O N S   O F   A L L   S P E C I E S"
1600 PRINT"I N   A   C O M P L E X   I O N   E Q U I L I B R I U M ."
1700 PRINT"Y O U   M U S T   T Y P E   I N   T H E   S T A B I L I T Y"
1800 PRINT"C O N S T A N T S .":PRINT:PRINT:PRINT
1850 PRINT:PRINT:PRINT
1900 REM IF YOU'RE USING A HARD-COPY TERMINAL YOU CAN OMIT THE
1910 REM SUBROUTINE AT LINE 64000.
1920 GOSUB 64000
```

Dear David Ahl:

I read the following interview with Lee Felsenstein on the Sol computer in the July-August 1977 *Creative Computing*.

> **Ahl:** Do you have some feeling concerning what people are doing with them so far?
>
> **Felsenstein:** It's very hard for me to tell... they're getting SOLs in order to do something. Just what it is we don't know. ....

In an effort to clear up some of the mystery I am sending you a sample of what I have been doing with my Sol omputer. I am a chemistry student nd computer fanatic. I bought the Sol ecause it comes assembled and it has ι keyboard instead of those ludicrous oggle switches!

In the Spring of 1976 I began working with Dr. L.A. Bares of Indiana University—Purdue University at Fort Wayne to build a series of interactive programs to assist Inorganic Chemistry students with their laboratory work. This group of programs was written in BASIC on a CDC-6600 and was used this Spring by chemistry students. The series presently has about ten programs. I am sending you one of these programs that I modified to run on my Sol.

This program finds the concentrations of the various species present in a complex-ion equilibrium. The user enters the stability constants and the analytical concentrations of the metal and ligand. The effects of pH and solubility on the equilibrium have been ignored for simplicity.

Our other Inorganic Chemistry programs assist the lab student as he/she finds the stability constants of a complex ion and then, with these constants known, proceed to plot beautiful logarithmic distribution graphs showing the relationship between ligand concentration and various complex ion concentrations.

We also wrote a program to do similar plots of polyprotic acid species versus pH, and a program to plot redox titration curves, just for fun.

The Sol-compatible versions of several of our other programs must wait until Processor Technology produces its long-awaited 8K BASIC. (I've been waiting since February!)

I believe the only non-standard BASIC statement I have used is: SET S = (a number). This changes the video display speed for Processor Technology's VDM. If you aren't using a Processor Technology computer, omit the statements about display speed.

For more information on our programs contact

J. P. Peer
417 West Water Street
Berne, Indiana 46711

and

Dr. L. A. Bares
Indiana University—Purdue University at Fort Wayne
2101 Coliseum Boulevard East
Fort Wayne, Indiana 46805

References

Angelici, R. J., *Synthesis & Technique in Inorganic Chemistry, Experiment* 13 (1977).

Blackburn, T. R., *Equilibrium* (1969).

Li, N. C., J. M. White & R. L. Yoest, *J. Amer. Chem. Soc.*, 78, 5218 (1956).

```
2000 FOR Z=1 TO 6:PRINT:NEXT Z
2100 PRINT"          August, 1977 by J. P. Peer"
2200 FOR Z=1 TO 7:PRINT:NEXT Z
2300 GOSUB 64000
2400 PRINT
2500 L=0:U=99
2600 PRINT"YOU CAN PICK THE DISPLAY SPEED FOR THIS PROGRAM."
2700 PRINT"'0' IS STANDARD (FAST DISPLAY), '99' IS VERY SLOW."
2800 INPUT"Type the display speed ( 0 to 99 ) here: ",S2
2900 A=S2
3000 GOSUB 60000
3050 REM CHANGE VDM DISPLAY SPEED TO USER'S WHIM.
3100 SET S=A
3200 PRINT"(clear key)"
3300 L=1:U=10
3400 PRINT:PRINT
3500 INPUT"Type the number of Stability Constants ( 1 to 10 ) here: ",L1
3510 REM IF YOUR BASIC WON'T ACCEPT THIS TYPE OF INPUT STATEMENT TRY
3520 REM THE FOLLOWING SUBSTITUTION:
3530 REM           PRINT"Type the number of ...... here: ";
3540 REM           INPUT A
3600 A=L1
3700 GOSUB 60000
3800 L1=INT(A)
4000 PRINT"(clear key)"
4100 PRINT:PRINT
4200 PRINT"   OKAY.  NOW TYPE IN THE STABILITY CONSTANTS."
4300 FOR I=1 TO L1
4400 PRINT"Type K(";I;" ) here: "; : INPUT" ",K(I)
4500 NEXT I
4600 PRINT
4700 L=0:U=50
4800 PRINT"NOW YOU NEED TO TYPE IN THE ANALYTICAL CONCENTRATIONS"
4900 PRINT"OF THE LIGAND AND METAL ( PLEASE USE MOLES / LITER )."
5000 INPUT"Type the Ligand concentration ( 0 to 50 ) here: ",L2
5100 A=L2
5200 GOSUB 60000
5300 L2=A
5400 INPUT"Type the Metal concentration ( 0 to 50 ) here: ",M
5500 A=M
5600 GOSUB 60000
5700 M=A
5800 REM FIND alpha 0.  SEE BLACKBURN'S 'EQUILIBRIUM' FOR THE THEORY.
5900 T=1:S=1:N=1
6000 T=T*K(N)*L2
6100 S=S+T
6200 N=N+1
6300 IF N <= L1 THEN 6000
6400 A1(0)=1/S
6500 REM FIND ALL THE OTHER alphas.
6600 N=1
6700 A1(N)=K(N)*A1(N-1)*L2
6800 N=N+1
6900 IF N <=L1 THEN 6700
7000 PRINT"(clear key)"
7100 PRINT
7200 PRINT"ALPHAS";TAB(27);"CONCENTRATIONS OF COMPLEXES"
7300 PRINT"++++++";TAB(27);"+++++++++++++++++++++++++++"
7400 FOR J=0 TO L1
7500 PRINT"#";J;" =";A1(J);TAB(21);J;"-Ligand complex =";A1(J)*M;
7550 PRINT" molar"
7600 NEXT J
7650 PRINT
7700 INPUT"To rerun this program type a zero here: ",Z
7800 IF Z ≠ 0 THEN STOP
7900 PRINT"(clear key)"
8000 INPUT"To use the same Stability Constants type zero here: ",Z
8100 IF Z = 0 THEN 4600
8200 GO TO 2400
59999 STOP
60000 REM MY WORLD-FAMOUS ERROR LIMITS SUBROUTINE.  LET'S NOT ALLOW
60010 REM THE USER TO MAKE MISTAKES WHEN S/HE TYPES IN DATA.  INPUT
60020 REM MUST BE WITHIN THE LIMITS OF 'L' AND 'U'.  'A' IS RETURNED.
60030 REM ANY GOOD BASIC INTERPRETER SHOULD HAVE THIS ABILITY BUILT
60040 REM IN AS AN INTRINSIC FUNCTION.
60100 IF A >= L THEN IF A <= U THEN RETURN
60200 PRINT"OOOOPS!  THIS NUMBER MUST BE FROM ";L;" TO ";U
60300 PRINT"PLEASE TRY AGAIN HERE: "; : INPUT" ",A
60400 GO TO 60100
63999 STOP
64000 REM SUBROUTINE TO STOP SCROLLING OF THE SCREEN.  LET'S HAVE
64010 REM PITY ON THE POOR USER WHO CAN'T READ AT 120000 BAUD AND
64020 REM ALLOW HIM/ER TO READ WHAT WE PRINTED BEFORE IT DISAPPEARS.
64030 REM A GOOD INTERPRETER WOULD HAVE THIS AS AN INTRINSIC FUNCTION
64040 REM ALSO.  FOR WORK WITH CRT'S IT'S INEVITABLE.
64100 INPUT"To continue, type any number here: ",Z
64200 PRINT"(clear key)"
64300 RETURN
```

Games

Another new game from Creative Computing....

OTHELLO

Ed Wright*

Language: FORTRAN
Description: Othello is a two-player game played on an 8-by-8 grid. The game begins with four chips on the board—two black and two white. In alternating moves, the players place a chip on the board (one player uses white chips, the other black chips). When, by making your move, you succeed in getting one or more chips of the opposing color between two of yours (in a horizontal, vertical, or diagonal row) they are converted to chips of your color. The game ends when the board is completely filled and the winner is determined on the basis of who has more chips of his color on the board. In the computerized version of the game presented here, you are pitted against the computer.

The author provided the following explanation of each subroutine:

Subroutine MOVEG (B, OC, NM, MOVESI, MOVESJ, DIR, LC, JAA, IAA IM, NØMVE, NFLIP): Generates all the legal moves from any possible board position. B is a 2D array that stores the present board position (100="off the board," 1=white, -1=black, 0=empty square). ØC is the "opposite-color variable" and tells the program what color the opponent is. If - ØC is put in the place of ØC, then the subroutine will generate all the legal moves for the opponent. MOVESI and MOVESJ are arrays that store the I and J coordinates of each of the legal moves returned from the subroutine. DIR is a 2D array that stores the number of the direction that pieces are flipped for each move. LC is an array that stores how many different directions pieces are flipped for each move. IM is the number of possible moves generated. NØMVE (could be eliminated and IM used) is a variable that if equal to 1 indicates no moves exist for the player from the present board position. NØMVE equal to 0 indicates the player does have at least one move. NFLIP is an array that stores how many pieces are flipped for each of the IM moves.

Subroutine CØUNT(B, ØC, NØC): Counts up the opponent's number of pieces on the board. If -ØC is put in place of ØC, then it counts up the number of pieces the computer has.

Subroutine BOARDP(B,NM): Prints out the board after having converted the 1's and 0's and -1's to W's, .'s, and B's. (See sample game included).

Subroutine BOARDC(MOVESI, MOVESJ, IF, IAA, JAA, B, ØC, DIR, LC): Changes the board for the IFth move of the possible moves stored in MOVESI, MOVESJ. Depending upon ØC, it too can either change the board for a move made by the computer or by its opponent.

Subroutine MOVEE(B, ØC, NM, MOVESI, MOVESJ, NFLIP, DIR, LC, IM, IF, IAA, JAA): Evaluates each of the moves given to it from the move-generator subroutine and returns what it considers to be the best move via the integer IF; that is, the move it selects is MOVESI(IF), MOVESJ(IF). I have not tried it, but I believe you could replace ØC with -ØC in this routine and set up the program to play itself and "fine-tune" the evaluation of the moves. The present version of MOVEE only looks ahead two ply and evaluates all the possible board positions. I am working on a version of MOVEE that will look ahead four ply (two turns) and use pruning techniques and a more general evaluation "function."

Final note: Originally, if the computer was allowed to go first, I had a random-number generator that chose one of the four possible opening moves for the computer to make. However, since the opening position is completely symmetric, it makes no difference which of these four opening moves one makes. Therefore, in the interest of reducing slightly the size of the program, I removed the random-number generator and associated statements and now just allow the program to make one of the four possible opening moves. I feel the savings in space is more important to the computer hobbyist.

Also, I am continually fine-tuning the move-evaluation subroutine that is included the printout, although its basic structure will not change.

Sample Run

—Game opening—

```
     BOARD POSITION AFTER   0 MOVES

J = 1  2  3  4  5  6  7  8
I
=
1   .  .  .  .  .  .  .  .
2   .  .  .  .  .  .  .  .
3   .  .  .  .  .  .  .  .
4   .  .  .  W  B  .  .  .
5   .  .  .  B  W  .  .  .
6   .  .  .  .  .  .  .  .
7   .  .  .  .  .  .  .  .
8   .  .  .  .  .  .  .  .


WELCOME TO THE GAME OF OTHELLO. DO YOU WISH TO
GO FIRST? YOU ARE WHITE IF YOU ARE FIRST.
?
'yes'

DO YOU WISH TO BE GIVEN A HANDICAP?
?
'no'

DO YOU WISH TO GIVE ME A HANDICAP?
?
'no'


WHAT IS YOUR MOVE ? (I,J).
? 00014
4,6

     BOARD POSITION AFTER   1 MOVES

J = 1  2  3  4  5  6  7  8
I
=
1   .  .  .  .  .  .  .  .
2   .  .  .  .  .  .  .  .
3   .  .  .  .  .  .  .  .
4   .  .  .  W  W  W  .  .
5   .  .  .  B  W  .  .  .
6   .  .  .  .  .  .  .  .
7   .  .  .  .  .  .  .  .
8   .  .  .  .  .  .  .  .

MY MOVE IS : 5,6
```

---

*802 Coriander Dr., Torrance, CA 90502.

—Later in the game—

```
WHAT IS YOUR MOVE ? (I,J).
? 00014
3,5

      BOARD POSITION AFTER  11 MOVES

J = 1  2  3  4  5  6  7  8
I
"

1   .  .  .  .  .  .  .  .
2   .  .  .  .  .  .  .  .
3   .  .  .  .  W  B  .  .
4   .  .  .  W  W  W  .  .
5   B  B  B  B  W  W  .  .
6   .  .  .  W  W  W  .  .
7   .  .  .  W  .  .  .  .
8   .  .  .  .  .  .  .  .

MY MOVE IS : 8,4

      BOARD POSITION AFTER  12 MOVES
```

—Still later in the game—

```
WHAT IS YOUR MOVE ? (I,J).
? 00014
3,8

      BOARD POSITION AFTER  23 MOVES

J = 1  2  3  4  5  6  7  8
I
"

1   .  .  .  .  .  .  .  .
2   .  .  W  .  .  .  .  .
3   .  .  W  W  W  W  W  W
4   .  .  W  B  W  B  W  W
5   B  B  W  B  B  W  B  .
6   .  .  W  B  W  W  .  B
7   .  .  .  B  .  .  .  .
8   .  .  .  B  .  .  .  .

MY MOVE IS : 2,4

      BOARD POSITION AFTER  24 MOVES
```

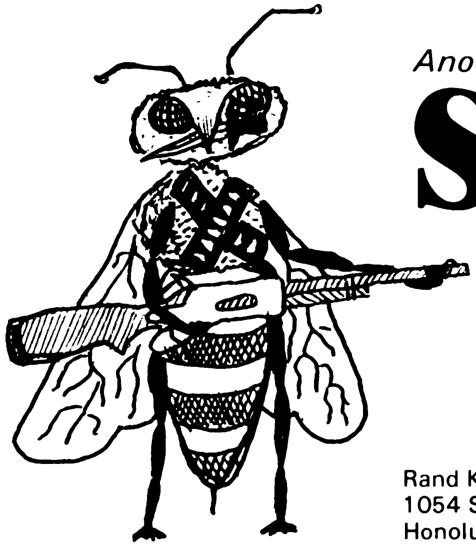
—At the end of the game—

```
WHAT IS YOUR MOVE ? (I,J).
? 00014
7,1

      BOARD POSITION AFTER  59 MOVES

J = 1  2  3  4  5  6  7  8
I
"

1   B  B  B  B  B  B  B  B
2   W  B  B  B  W  W  .  B
3   W  B  B  B  W  W  B  B
4   W  B  B  B  W  B  W  B
5   W  B  W  W  B  W  W  B
6   W  W  W  W  W  B  W  B
7   W  W  W  W  W  W  W  B
8   B  B  B  B  B  B  B  B

MY MOVE IS : 2,7

      BOARD POSITION AFTER  60 MOVES
```

Program Listing

```
      INTEGER B(10,10),DIR(30,8)
      DIMENSION DRSPON(2),IAA(8),JAA(8),MOVESI(30)
     2,MOVESJ(30),LC(30),NFLIP(30)
      DATA DRSPON /'YES','NO'/
      DATA IAA /-1,-1,-1,0,1,1,1,0/
      DATA JAA /-1,0,1,1,1,0,-1,-1/
      COMMON /TWO/ OC
   22 DO 10 I=1,10
      DO 10 J=1,10
      B(I,J)=0
      IF(I.EQ.1.OR.I.EQ.10)B(I,J)=100
   10 IF(J.EQ.1.OR.J.EQ.10)B(I,J)=100
      B(5,5)=1
      B(5,6)=-1
      B(6,5)=-1
      B(6,6)=1
      CALL BOARDP(B,0,0)
      WRITE(6,601)
  601 FORMAT(//,1X,'WELCOME TO THE GAME OF OTHELLO. DO YOU WISH TO
     2',/,' GO FIRST? YOU ARE WHITE IF YOU ARE FIRST.')
      READ(5,*)RESPON
      OC=1
      IF(RESPON.EQ.DRSPON(2))GO TO 11
      CALL HANDIC(OC,B,DRSPON,NHD)
      NM=NHD
    8 IF(NM.EQ.60)GO TO 15
      CALL MOVEG(B,-OC,NM,MOVESI,MOVESJ,DIR,LC,JAA,IAA,
     2IM,NOMVE,NFLIP)
      IF(IM.EQ.0)GO TO 12
      WRITE(6,713)
  713 FORMAT(//,1X,'WHAT IS YOUR MOVE ? (I,J).')
   14 READ(5,*)MOVEI,MOVEJ
      MOVEJ=MOVEJ+1
      MOVEI=MOVEI+1
      DO 9 I=1,IM
      IF(MOVESI(I).EQ.MOVEI.AND.MOVESJ(I).EQ.MOVEJ)GO TO 13
    9 CONTINUE
      WRITE(6,701)
  701 FORMAT(1X,'MOVE INVALID. PLEASE RE-ENTER')
      GO TO 14
   13 NM=NM+1
      CALL BOARDC(MOVESI,MOVESJ,I,IAA,JAA,B,-OC,DIR,LC)
      CALL BOARDP(B,NM,NHD)
      GO TO 2
   11 OC=-1
      CALL HANDIC(OC,B,DRSPON,NHD)
      B(5,7)=1
      B(5,6)=1
      NM=NHD+1
      CALL BOARDP(B,NM,NHD)
      GO TO 8
   12 WRITE(6,756)
  756 FORMAT(/,1X,'I CAN SEE NO MOVE FOR YOU , SO I WILL ',
     2'MOVE IF I CAN.')
    2 IF(NM.EQ.60)GO TO 15
      CALL MOVEG(B,OC,NM,MOVESI,MOVESJ,DIR,LC,JAA,IAA,
     2IM,NOMVE,NFLIP)
      IF(IM.EQ.0)GO TO 20
      CALL MOVEE(B,OC,NM,MOVESI,MOVESJ,NFLIP,DIR,LC,IM,IF,IAA,JAA)
      MOVEI=MOVESI(IF)-1
      MOVEJ=MOVESJ(IF)-1
      WRITE(6,603)MOVEI,MOVEJ
  603 FORMAT(/,1X,'MY MOVE IS : ',I1,',',I1)
      CALL BOARDC(MOVESI,MOVESJ,IF,IAA,JAA,B,OC,DIR,LC)
      NM=NM+1
      CALL BOARDP(B,NM,NHD)
      GO TO 8
   20 WRITE(6,602)
  602 FORMAT(/,1X,'DO YOU HAVE A MOVE?')
      READ(5,*)RESPON
      IF(RESPON.EQ.DRSPON(1))GO TO 8
      IF(IM.NE.0)GO TO 2
   15 CALL COUNT(B,OC,NOC)
      CALL COUNT(B,-OC,NC)
      IF(NOC.GT.NC)WRITE(6,610)
  610 FORMAT(/,1X,'CONGRATULATIONS,YOU PLAYED WELL AND HAVE WON.',
     2/,1X,'THANK YOU FOR A FINE GAME')
      IF(NOC .LT. NC)WRITE(6,611)
  611 FORMAT(/,1X,'YOU PLAYED WELL ; HOWEVER,YOUR LUCK WAS BAD AND'

     2,/,'I HAVE WON. THANK YOU FOR A FINE GAME.')
      IF(NOC .EQ. NC)WRITE(6,612)
  612 FORMAT(/,'YOU PLAYED WELL AND WE HAVE TIED. I WAS LUCKY.',/,
     2'THANK YOU FOR A FINE GAME.')
      WRITE(6,613)
  613 FORMAT(/,1X,'DO YOU WISH TO PLAY AGAIN?')
      READ(5,*)RESPON
      IF(RESPON.EQ.DRSPON(1))GO TO 22
      STOP
      END
      SUBROUTINE MOVEG(B,OC,NM,MOVESI,MOVESJ,DIR,LC,JAA,IAA,IM,
     2NOMVE,NFLIP)
      INTEGER B(10,10),DIR(30,8)
      DIMENSION MOVESI(30),MOVESJ(30),LC(30),NFLIP(30),
     2IAA(1),JAA(1)
      COMMON /TWO/ OCA
      DO 1 I=1,30
      LC(I)=0
    1 NFLIP(I)=0
      IM=0
      DO 20 I=2,9
      DO 20 J=2,9
      IF(B(I,J))20,3,20
    3 IC=0
      DO 5 L=1,8
      IA=IAA(L)
      JA=JAA(L)
      IF(B(I+IA,J+JA)-OC)5,6,5
    6 IV=1
    4 IV=IV+1
      MVI=I+IV*IA
      MVJ=J+IV*JA
      IF(B(MVI,MVJ))7,5,7
    7 IF(B(MVI,MVJ)-100)11,5,11
   11 IF(B(MVI,MVJ)-OC)8,4,8
    8 IF(IC-1)13,12,13
   13 IM=IM+1
      IC=1
   12 NFLIP(IM)=NFLIP(IM)+IV
      LC(IM)=LC(IM)+1
      LD=LC(IM)
      DIR(IM,LD)=L
    5 CONTINUE
      IF(IC)21,20,21
   21 MOVESI(IM)=I
      MOVESJ(IM)=J
   20 CONTINUE
      IF(IM)31,31,30
   31 IF(OCA.EQ.OC)WRITE(6,100)
  100 FORMAT(/,1X,'I HAVE NO MOVE AND MUST PASS')
   30 RETURN
      END
      SUBROUTINE COUNT(B,OC,NOC)
      INTEGER B(10,1)
      NOC=0
      DO 10 I=2,9
      DO 10 J=2,9
      IF(B(I,J)-OC)10,5,10
    5 NOC=NOC+1
   10 CONTINUE
      RETURN
      END
      SUBROUTINE BOARDP(B,NM,NHD)
      DIMENSION OUT(3),POUT(10,10)
      INTEGER B(10,1)
      DATA OUT /'B','.','W'/
      NMP=NM-NHD
      WRITE(6,100)NMP
  100 FORMAT(/,6X,'BOARD POSITION AFTER  ',I2,' MOVES',/)
      WRITE(6,101)
  101 FORMAT(' J = 1  2  3  4  5  6  7  8')
      WRITE(6,102)
  102 FORMAT(' I')
      WRITE(6,103)
  103 FORMAT(' "')
      DO 9 I=2,9
      DO 9 J=2,9
      IS=B(I,J)+2
    9 POUT(I,J)=OUT(IS)
      DO 10 I=2,9
      I1=I-1
   10 WRITE(6,104)I1,(POUT(I,J),J=2,9)
  104 FORMAT(1X,I1,3X,8(A1,2X))
```

```
      RETURN
      END
      SUBROUTINE BOARDC(MOVESI,MOVESJ,IF,IAA,JAA,B,OC,DIR,LC)
      INTEGER B(10,10),DIR(30,8)
      DIMENSION MOVESI(30),MOVESJ(30),IAA(1),JAA(1),LC(30)
      MI=MOVESI(IF)
      MJ=MOVESJ(IF)
      B(MI,MJ)=-OC
      NDIR=LC(IF)
      DO 40 I=1,NDIR
      L=DIR(IF,I)
      IA=IAA(L)
      JA=JAA(L)
      IV=0
   31 IV=IV+1
      MVI=MI+IV*IA
      MVJ=MJ+IV*JA
      IF(B(MVI,MVJ)+OC)39,40,39
   39 B(MVI,MVJ)=-OC
      GO TO 31
   40 CONTINUE
      RETURN
      END
      SUBROUTINE MOVEE(B,OC,NM,MOVESI,MOVESJ,NFLIP,DIR,LC,
     2IM,IF,IAA,JAA)
      INTEGER B(10,1),DIR(30,1),BT(10,10),BTT(10,10),DIRB(20,8)
     2,BTTS(9,9,20),DIRBB(20,8)
      DIMENSION MOVESI(1),MOVESJ(1),LC(1),NFLIP(1),MBI(20),MBJ(20)
     2,LCB(20),NFLIPB(30),IAA(1),JAA(1),IY(24),JY(24),
     2IMID(24),JMID(24),ID(24),JD(24),NCORNI(4),NCORNJ(4)
     2,MBBI(20),MBBJ(20),LCBB(20),NFLIB(30)
      DATA NCORNI,NCORNJ /2,2,9,9,2,9,9,2/
      DATA ID,JD /3,4,5,6,7,8,6*9,8,7,6,5,4,3,12*2,3,4,5,6,7,8,
     26*9,8,7,6,5,4,3/
      DATA IY,JY /5,1,3,8,1,6,9,1,9,9,1,9,6,1,8,3,1,5,2,1,2,2,1,2,
     22,1,2,2,1,2,5,1,3,8,1,6,9,1,9,9,1,9,6,1,8,3,1,5/
      DATA IMID,JMID /4,1,4,7,1,7,9,1,9,9,1,9,7,1,7,4,1,4,2,1,2,2,
     21,2,2,1,2,2,1,2,4,1,4,7,1,7,9,1,9,9,1,9,7,1,7,4,1,4/
      ICO=0
      IF=1
      IF(NM.EQ.59)GO TO 20
   10 DO 12 I=1,IM
      MI=MOVESI(I)
      MJ=MOVESJ(I)
      IF(MI.NE.3.AND.MI.NE.8)GO TO 13
      IF(MJ.NE.3.AND.MJ.NE.8)GO TO 13
      IF(MI.EQ.3.AND.MJ.EQ.3)IC=1
      IF(MI.EQ.3.AND.MJ.EQ.8)IC=2
      IF(MI.EQ.8.AND.MJ.EQ.8)IC=3
      IF(MI.EQ.8.AND.MJ.EQ.3)IC=4
      IF(B(NCORNI(IC),NCORNJ(IC)).EQ.0)NFLIP(I)=NFLIP(I)-50
   13 IF(MI.NE.2.AND.MI.NE.9)GO TO 11
      IF(MJ.NE.2.AND.MJ.NE.9)GO TO 11
      ICO=ICO+1
      NFLIP(I)=NFLIP(I)+60
   11 IF(MI.LE.3.OR.MI.GE.8)GO TO 2
      IF(MJ.LE.3.OR.MJ.GE.8)GO TO 2
      NFLIP(I)=NFLIP(I)+10
      GO TO 12
    2 ND=LC(I)
      DO 5 J=1,ND
      L=DIR(I,J)
      IA=IAA(L)
      JA=JAA(L)
      IV=1
    4 IV=IV+1
      MVI=MI+IV*IA
      MVJ=MJ+IV*JA
      IF(B(MVI,MVJ)-OC)6,4,6
    6 IV=IV+1
      MVI=MI+IV*IA
      MVJ=MJ+IV*JA
      IF(B(MVI,MVJ)-OC)7,8,7
    7 IF(B(MVI,MVJ)+OC)5,6,5
    8 IF(B(MI-IA,MJ-JA))5,9,5
    9 NFLIP(I)=NFLIP(I)-5
      GO TO 12
    5 CONTINUE
   12 CONTINUE
      DO 32 I=1,IM
      NSUBO=0
      MI=MOVESI(I)
      MJ=MOVESJ(I)
      IC=0
      DO 33 K=1,10
      DO 33 J=1,10
   33 BT(K,J)=B(K,J)
      LL=0
      DO 56 J=1,24
      IPP=ID(J)
      JPP=JD(J)
      IF(MOVESI(I).NE.IPP.OR.MOVESJ(I).NE.JPP)GO TO 56
      LL=J
   56 CONTINUE
      CALL BOARDC(MOVESI,MOVESJ,I,IAA,JAA,BT,OC,DIR,LC)
      CALL MOVEG(BT,-OC,NM,MBI,MBJ,DIRB,LCB,JAA,IAA,IM1,
     2NOMVE,NFLIPB)
      IF(IM1.NE.0)GO TO 63
      NFLIP(I)=NFLIP(I)+100
      GO TO 32
   63 DO 36 J=1,IM1
      DO 34 K=1,10
      DO 34 L=1,10
   34 BTT(K,L)=BT(K,L)
      CALL BOARDC(MBI,MBJ,J,IAA,JAA,BTT,-OC,DIRB,LCB)
      IF(LL.EQ.0)GO TO 38
      IC=1
      IZ=IY(LL)
      JZ=JY(LL)
      IF(B(IZ,JZ)+OC)41,40,41
   40 MK=JMID(LL)
      ML=IMID(LL)
      IF(B(ML,MK).EQ.0)NSUBO=90
   41 IF(BTT(MI,MJ).NE.OC)GO TO 38
      NFLIP(I)=NFLIP(I)-40
      IC=2
   38 CONTINUE
      CALL COUNT(BTT,-OC,NOC)
      IF(NOC)39,39,42
   39 NFLIP(I)=NFLIP(I)-200
      GO TO 32
   42 DO 37 K1=2,9
      DO 37 K2=2,9
   37 BTTS(K1,K2,J)=BTT(K1,K2)
      DO 100 IL=2,9
      DO 100 JL=2,9
      IF(BTT(IL,JL))96,100,96
   96 IF(BTT(IL,JL) - OC)99,100,99
   99 DO 90 IZ=1,8
      IV=0
   80 IV=IV+1
      ILL=IL+IV*IAA(IZ)
      JLL=JL+IV*JAA(IZ)
      IF(BTT(ILL,JLL))98,36,98
   98 IF(BTT(ILL,JLL)-100)97,36,97
   97 IF(BTT(ILL,JLL)-OC)80,90,80
   90 CONTINUE
  100 CONTINUE
   95 CALL MOVEG(BTT,OC,NM,MBBI,MBBJ,DIRBB,LCBB,JAA,IAA,IM2,
     2NOMVE,NFLIB)
      IF(IM2.EQ.0)GO TO 103
      DO 102 IL=1,IM2
      IF(MBBI(IL).NE.2.OR.MBBI(IL).NE.9)GO TO 102
      IF(MBBJ(IL).NE.2.OR.MBBJ(IL).NE.9)GO TO 102
      GO TO 36
  102 CONTINUE
  103 NFLIP(I)=NFLIP(I)-190
   36 CONTINUE
      IF(IC.NE.1)GO TO 35
      DO 50 K=1,24
      IQ=ID(K)
      JQ=JD(K)
      IF(MI.EQ.IQ.AND.MJ.EQ.JQ)GO TO 50
   52 IF(B(IQ,JQ)+OC)50,53,50
   53 DO 54 K1=1,IM1
   54 IF(BTTS(IQ,JQ,K1).EQ.OC)NFLIP(I)=NFLIP(I)-8
   50 CONTINUE
      NFLIP(I)=NFLIP(I)+25-NSUBO
   35 DO 60 K=1,4
      KC1=NCORNI(K)
      KC2=NCORNJ(K)
      IF(B(KC1,KC2))60,58,60
   58 DO 61 K1=1,IM1
   61 IF(BTTS(KC1,KC2,K1).EQ.OC)NFLIP(I)=NFLIP(I)-55
      IF(ICO.LE.1)GO TO 60
      IF(MI.EQ.KC1.AND.MJ.EQ.KC2)GO TO 66
      DO 62 K1=1,IM1
   62 IF(BTTS(KC1,KC2,K1).EQ.OC)NFLIP(I)=NFLIP(I)-20
   60 CONTINUE
   32 CONTINUE
      NFLIPM=-800
      DO 15 I=1,IM
      IF(NFLIP(I)-NFLIPM)15,16,16
   16 NFLIPM=NFLIP(I)
      IF=I
   15 CONTINUE
   20 RETURN
      END
      SUBROUTINE HANDIC(OC,B,DRSPON,NHD)
      DIMENSION DRSPON(1)
      INTEGER B(10,1)
      NHD=0
      WRITE(6,608)
  608 FORMAT(/,1X,'DO YOU WISH TO BE GIVEN A HANDICAP?')
      READ(5,*)RESPON
      IF(RESPON.EQ.DRSPON(1))GO TO 7
      WRITE(6,610)
  610 FORMAT(/,1X,'DO YOU WISH TO GIVE ME A HANDICAP?')
      READ(5,*)RESPON
      IF(RESPON.EQ.DRSPON(2))GO TO 100
      NAH=-OC
      WRITE(6,609)
  609 FORMAT(/,1X,'HOW MANY CORNERS? (1-4)')
  607 READ(5,*)NHD
      IF(NHD.LT.1.OR.NHD.GT.4)GO TO 607
      CALL HANDI(B,NHD,NAH,OC)
      CALL BOARDP(B,0,0)
      GO TO 100
    7 NAH=OC
      WRITE(6,609)
  606 READ(5,*)NHD
      IF(NHD.LT.1.OR.NHD.GT.4)GO TO 606
      CALL HANDI(B,NHD,NAH,OC)
      CALL BOARDP(B,0,0)
  100 RETURN
      END
      SUBROUTINE HANDI(B,NHD,NAH,OC)
      INTEGER B(10,1)
      INTEGER NCORNI(4),NCORNJ(4)
      DATA NCORNI,NCORNJ /2,2,9,9,2,9,9,2/
      SIGN=-1.
      IF(NAH.EQ.OC)SIGN=+1.
      DO 10 I=1,NHD
      I1=NCORNI(I)
      I2=NCORNJ(I)
   10 B(I1,I2)=SIGN*OC
      RETURN
      END
```



Another new game from Creative Computing . . . .

SWARMS

Rand K. Miller
1054 S. King St.
Honolulu, HI 96814

Language: BASIC (DEC 10)
Description: SWARMS is a computer simulation (fancy word for game) that was conceived from the book *The Swarm* by Arthur Herzog. The program puts you in charge of the defense of the entire United States when swarms of ferocious South American hybrid bees suddenly start appearing in different sections of the country. The program is provided with in depth instructions which explain the situation very thoroughly.

My goals for this game were to create a program that was 1) my own creation, not an improvement in somebody else's game, 2) as realistic as possible, 3) not another Star Trek game. Since I wanted this project to be a test of my programming skills, I wrote the instructions first and made the game to follow the instruction as closely as possible. The instructions were updated at the end but the program ended up pretty much what I had planned.

I would like to give special credit to the Albuquerque Public School System who own and operate a DEC-10 computer system just for high school students. Their set up is slightly ahead of the times.

I would also like to give no credit to the Honolulu Public School System which owns and operates nothing for high school students. Their entire city is years behind the times.

NOTE: When adapting this program for other computer systems, special attention should be paid to the "tab" and "print using" statements (5100-5250) and the margin statement (90) which is not necessarily needed.

Rand Miller at home with a Friend.

```
DO YOU NEED INSTRUCTIONS ?NO
TIME:    1
ENTER YOUR LAST NAME FOR IDENTIFICATION CHECK.
 ?MILLER
ENTER YOUR CODE WORD FOR LATER VARIFICATION.
 ?SWARMS

COMMAND ?1
TIME:    2
1)       ATTACK SCAN MAP
--------------------------------------------------------------------
                                                  ---
                      CANADA: NO INFORMATION     /  /
                 -----------------------        /  /
                 1   1   1   1    1    1\_    _/  /
                 1   1   1   1 *  1    1  \__/---/
                 1   1___1___1____1____1   1    /
                 1___1   1   1    1    1   1   1
                 1   1   1   1    1    1___1___1
                 1   1___1___1____1____1   1   1
                  \  1   1   1    1    1   1   1
                   \ 1   1   1    1    1   1___/
                    \1___1___1____1____1___1 *1
                             \_   1         \ 1
                               \  1         1 1
                                \_1         1_1
                      MEXICO: NO INFORMATION
--------------------------------------------------------------------


COMMAND ?2
TIME:    3
2)       ETA REPORT

SECTION ?9
*****************************************
THE BEES WILL ARRIVE AT THE MAJOR CITY IN
        SECTION 9 AT 9 HOURS.
*****************************************

COMMAND ?2
TIME:    4
2)       ETA REPORT

SECTION ?21
*****************************************
THE BEES WILL ARRIVE AT THE MAJOR CITY IN
        SECTION 21 AT 10 HOURS.
*****************************************
*THE SWARM IN SECTION 9 HAS SPREAD TO SECTION 8

COMMAND ?3
TIME:    5

3)       BATTLE PHASE OPTIONS
SECTION ?8
PHASE ?3
PROJECT BRUSH FIRE: PHASE THREE, NOW BEING ATTEMPTED.
*THE PHASE ON SECTION 8 WAS SUCCESSFUL_ __ __ __ __ __ __ _
*THE SWARM IN SECTION 8 IS READY TO BE DESTROYED
*THE SWARM IN SECTION 9 HAS SPREAD TO SECTION 10
*THE SWARM IN SECTION 21 HAS SPREAD TO SECTION 20

COMMAND ?3
TIME:    6

3)       BATTLE PHASE OPTIONS
SECTION ?20
PHASE ?3
PROJECT BRUSH FIRE: PHASE THREE, NOW BEING ATTEMPTED.

COMMAND ?3
TIME:    7

3)       BATTLE PHASE OPTIONS
SECTION ?10
PHASE ?3
PROJECT BRUSH FIRE: PHASE THREE, NOW BEING ATTEMPTED.
*THE PHASE ON SECTION 10 WAS SUCCESSFUL

COMMAND ?3
TIME:    8
```

```
00010   PRINT"DO YOU NEED INSTRUCTIONS";
00020   INPUT I$
00030   PRINT"TIME:    1"
00040   PRINT"ENTER YOUR LAST NAME FOR IDENTIFICATION CHECK."
00050   INPUT N$
00060   PRINT"ENTER YOUR CODE WORD FOR LATER VARIFICATION."
00070   INPUT C$
00080   IF I$<>"YES" THEN 3570
00090   MARGIN 100
00100   PRINT
00110   PRINT
00120   PRINT
00130   PRINT"          IN 1956, APICULTURIST  WARWIC  KERR BROUGHT TWENTY AFRICAN"
00140   PRINT"QUEEN BEES INTO BRAZIL.    THOUGH IT IS KNOWN TO BE FEROCIOUS,  THE"
00150   PRINT"AFRICAN BEE PRODUCES 30  PERCENT MORE HONEY THAN EITHER THE ITALIAN"
00160   PRINT"OR GERMAN BEE THAT LONG DOMINATED BRAZILIAN BEEKEEPING.  KERR PLAN-"
00170   PRINT"NED TO CROSS-BREED HIS AFRICANS TO PRODUCE A MORE GENTLE BEE.  WHAT"
00180   PRINT"HE OBTAINED, HOWEVER, WAS A BEE WITH A DISPOSITION SO NASTY THAT IT"
00190   PRINT"THREATENED THE LIVES  AND LIVELIHOOD  OF ALMOST EVERY  BEEKEEPER IN"
00200   PRINT"8 STATES OF BRAZIL.  IN JUNE 1968, THE AGRARIAN DEFENSE DERECTORATE"
00210   PRINT"IN LIMA,  PERU, BEGAN EMERGENCY  MEASURES TO PREVENT AN INVASION OF"
00220   PRINT"THE DEADLY AFRICAN BEES FROM BRAZIL.  THE BEES QUICKLY SPREAD OVER"
00230   PRINT"AN AREA ABOUT  EQUAL TO THE SIZE OF THE CONTINENTAL  UNITED STATES."
00240   PRINT"SWARMS OF THESE FEROCIOUS AFRICAN HONEY BEES ARE NOW  MOVING TOWARD"
00250   PRINT"THE UNITED STATES AT THE RATE OF ABOUT 200 MILES A YEAR,  AND THERE"
00260   PRINT"SEEMS TO BE NO NATURAL  BARRIER TO BLOCK THEM, ACCORDING TO A STUDY"
00270   PRINT"FINANCED BY THE DEPARTMENT OF AGRICULTURE.  THIS GAME IS A SIMULA-"
00280   PRINT"TION OF  WHAT MIGHT HAPPEN  WHEN AND IF THE  BEES REACH THE  UNITED"
00290   PRINT"STATES.  IN THIS GAME ONE PERSON IS IN CHARGE OF THE DEFENSE OF THE"
00300   PRINT"ENTIRE UNITED STATES.  THAT ONE PERSON, THROUGH THE USE OF A COMPU-"
00310   PRINT"TER, HAS TO KEEP TRACK OF THE BEES AND TRY TO COMBAT THEM.   COMPU-"
00320   PRINT"TER RADAR  SUBSTATIONS ARE  POSITIONED IN 21 STRATEGIC AREAS OF THE"
00330   PRINT"UNITED  STATES TO SHOW ANY  SWARMS IN THOSE AREAS.  THE SWARMS CAN"
00340   PRINT"AND  WILL GROW AND SPREAD IF THEY ARE NOT COMBATTED.  THEY ARE ALSO"
00350   PRINT"CONSTANTLY  MOVING TOWARD MAJOR CITIES.   EACH SECTION HAS ONE CITY"
00360   PRINT"THAT IS CONSIDERED  MAJOR.   THE SIZE OF THE  SWARMS ARE SHOWN ON A"
00370   PRINT"MAP AND THE ESTIMATED TIME OF ARRIVAL (ETA) OF THE BEES TO A  MAJOR"
00380   PRINT"CITY IS  PRINTED OUT IN CHART FORM.   THE OBJECT OF THIS GAME IS TO"
00390   PRINT"DESTROY ALL OF THE SWARMS WITH AS FEW CASUALTIES AS POSSIBLE.   ALL"
00400   PRINT"OF THE COMPUTER  COMMANDS THAT CAN BE USED WILL NOW BE  PRINTED OUT"
00410   PRINT"ALONG WITH A DETAILED EXPLANATION OF EACH ONE.    IF YOU WANT SHORT"
00420   PRINT"INSTRUCTIONS THE NEXT TIME YOU PLAY ENTER 'SHORT' WHEN ASKED IF YOU"
00430   PRINT"WANT INSTRUCTIONS."
00440   PRINT
00450   PRINT
00460   PRINT
00470   PRINT
00480   PRINT
00490   PRINT
00500   PRINT
00510   PRINT
00520   PRINT"                          COMMANDS"
00530   PRINT"############################################################"
00540   PRINT
00550   PRINT"COMMAND 1:     ATTACK SCAN MAP"
00560   PRINT
00570   PRINT"          THIS COMMAND PRINTS OUT THE MAP OF THE U.S. AND THE"
00580   PRINT"          BEE ATTACKS.  HERE IS HOW THE SECTIONS ARE LAID OUT."
00590   PRINT
00600   PRINT
00610   PRINT
00620   PRINT
00630   PRINT"                                                        "
00640   PRINT"                         CANADA: NO INFORMATION     /---"
00650   PRINT"                    +---+---+---+----+----+        /18/"
00660   PRINT"                    I   I   I   I    I    I\      /  /"
00670   PRINT"                    I 1 I 3 I 6 I 9  I 13 I -\   /  /"
00680   PRINT"                    I   I---I---I----I----I16\+-/=--/"
00690   PRINT"                    I---I   I   I    I    I   I 19I"
00700   PRINT"                    I   I 4 I 7 I 10 I 14 I---I---I"
00710   PRINT"                    I 2 I---I---I----I----I   I   I"
00720   PRINT"                     \  I   I   I    I    I17 I 20I"
00730   PRINT"                      \ I 5 I 8 I 11 I 15 I   I   /"
00740   PRINT"                       \I---I---I----I----I---I21I "
00750   PRINT"                                \-   I         \ I"
00760   PRINT"                                  \12I         I I"
00770   PRINT"                                   \ I         I I"
00780   PRINT"                         MEXICO: NO INFORMATION -I"
00790   PRINT
00800   PRINT"                     ----------------------------------"
00810   PRINT
00820   PRINT"          THIS IS WHAT YOU MIGHT SEE ON AN ATTACK SCAN MAP."
00830   PRINT
00840   PRINT"                                                        "
00850   PRINT"                         CANADA: NO INFORMATION     /---"
00860   PRINT"                    +---+---+---+----+----+        /  /"
00870   PRINT"                    I   I   I   I    I    I\      /  /"
00880   PRINT"                    I   I   I $ I    I    I -\   /  /"
00890   PRINT"                    I   I---I---I----I----I +\+-/=--/"
00900   PRINT"                    I---I   I   I    I    I   I . I"
00910   PRINT"                    I   I + I   I    I    I---I---I"
00920   PRINT"                    I   I---I---I----I----I   I   I"
00930   PRINT"                     \  I   I   I    I    I * I   I"
00940   PRINT"                      \ I   I   I    I    I   I   /"
00950   PRINT"                       \I---I---I--$-I----I---I--I "
00960   PRINT"                                \-   I         \ I"
00970   PRINT"                                  \  I         I I"
00980   PRINT"                                   \ I         I I"
00990   PRINT"                         MEXICO: NO INFORMATION -I"
```

```
01990   PRINT"*PHASE THREE:"
02000   PRINT"              PROJECT BRUSH FIRE-                    1 HOUR"
02010   PRINT"--------------------------------------------------------------"
02020   PRINT"              LINE-BEARING WORKERS"
02030   PRINT"              STIMULATE AGGRESSION"
02040   PRINT
02050   PRINT"     HUNDREDS OF MILLIONS OF ARTIFICIALLY-REARED WORKERS (KEYED"
02060   PRINT"WITH AN ATTACK PHEROMONE) ARE DROPPED ON INFESTED AREAS.  THEY ARE"
02070   PRINT"CHEMICALLY COMPELLED TO SEARCH FOR AND ATTACK ANY AFRICAN BEEHIVE"
02080   PRINT"IN THE VICINITY CAUSING REDUCTION OF THE AFRICAN POPULATION."
02090   PRINT
02100   PRINT"++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++"
02110   PRINT
02120   PRINT"*PHASE FOUR:"
02130   PRINT"              PROJECT STERILE MALE-                  4 HOURS"
02140   PRINT"--------------------------------------------------------------"
02150   PRINT"              LINE-BEARING DRONES"
02160   PRINT"              STERILIZATION"
02170   PRINT
02180   PRINT"     HUNDREDS OF MILLIONS OF ARTIFICIALLY-REARED DRONES WOULD BE"
02190   PRINT"STERILIZED AND DROPPED FROM THE AIR ON INFESTED AREAS.  MATING WITH"
02200   PRINT"THE AFRICAN QUEENS, THEY WOULD INSERT USELESS SPERM TO FILL UP THE"
02210   PRINT"QUEEN.  THE UNFERTILIZED QUEEN WOULD THEN ONLY PRODUCE DRONE EGGS."
02220   PRINT
02230   PRINT"++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++"
02240   PRINT
02250   PRINT"*PHASE FIVE:"
02260   PRINT"              DESTRUCTION-                           3 HOURS"
02270   PRINT"--------------------------------------------------------------"
02280   PRINT"              BEE WARDENS"
02290   PRINT
02300   PRINT"     THIS DEFENSE IS TO BE USED ONLY AFTER THE BEE POPULATION"
02310   PRINT"HAS BEEN REDUCED BY ONE OF THE LOWER NUMBER PHASES.  TRAINED BEE"
02320   PRINT"WARDENS WILL ENTER THE FOREST IN A LIMITED AREA AND FINISH OFF"
02330   PRINT"THE BEES THAT ARE LEFT."
02340   PRINT"++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++"
02350   PRINT
02360   PRINT"*PHASE SIX:"
02370   PRINT"              URBAN DEFENSES-                        1 HOUR EACH"
02380   PRINT"--------------------------------------------------------------"
02390   PRINT"              ISOLATION DEFENSES"
02400   PRINT"              FINAL RESORT"
02410   PRINT
02420   PRINT"     A) FLIGHT PATTERNS"
02430   PRINT"              COATED METAL FOIL STRIPS ARE DROPPED ON SWARM TO SET"
02440   PRINT"              UP VIBRATIONS IN THE AIR TO DISTURB BEES FLIGHT PAT-"
02450   PRINT"              TERNS."
02460   PRINT"     B) SONIC BOOM"
02470   PRINT"              F-111A'S FLY THROUGH SWARM OF BEES AT SONIC SPEEDS"
02480   PRINT"              TO COMBINE SONIC BOOM,JET EXHAUST,AND JET STREAM."
02490   PRINT"     C)SUPER-SONIC BEAMS"
02500   PRINT"              SUPER-SONIC BEAM SYNTHESIZER, ORIGINALLY MEANT TO CONFUSE"
02510   PRINT"              THE GUIDANCE SYSTEMS OF SAM MISSILES, ARE PLACED IN STRA-"
02520   PRINT"              TEGIC AREAS AROUND THE CITY.  THE BEES WON'T BE KILLED"
02530   PRINT"              BUT THEIR SOLAR AND GRAVITATIONAL ORIENTATION SHOULD BE"
02540   PRINT"              DISRUPTED ENOUGH TO CAUSE THEM TO FLY INTO BUILDINGS OR"
02550   PRINT"              DIVE INTO THE GROUND."
02560   PRINT"     D)POLLUTION"
02570   PRINT"              THOUSANDS OF CARS ARE POSITIONED ALONG MAJOR TRAFFIC"
02580   PRINT"              ARTERIES OF THE CITY.  THEY ARE WIRED TO BE STARTED BY"
02590   PRINT"              REMOTE CONTROL.  THE BEES SHOULD DIE FROM THE POLLUTION."
02600   PRINT"     E)METHYL PARATHION"
02610   PRINT"              SMALL CROP DUSTERS ARE SENT UP TO SPRAY A METHYL PARA-"
02620   PRINT"              THION BARRIER IN THE PATH OF THE BEES."
02630   PRINT"     F)FIRE WALL"
02640   PRINT"              KEROSENE FILLED TANKER TRUCKS ARE IGNITED IN THE PATH OF"
02650   PRINT"              THE BEES.  SOME BEES SHOULD BE BURNED AND OTHERS SHOULD"
02660   PRINT"              BE CARRIED UP BY THE HOT AIR TO ALTITUDES BEYOND THEIR"
02670   PRINT"              CAPABILITIES."
02680   PRINT"     G)STROBE LIGHT"
02690   PRINT"              A HIGH-INTENSITY STROBE LIGHT IS POSITIONED ON TOP OF A"
02700   PRINT"              HIGH BUILDING.  IT IS SURROUNDED BY A HIGH-VOLTAGE WIRE"
02710   PRINT"              MESH.  THE BEES SHOULD BE ATTRACTED TO THE STROBE AND"
02720   PRINT"              ELECTROCUTED."
02730   PRINT
02740   PRINT"##############################################################"
02750   PRINT
02760   PRINT"COMMAND 4:          EVACUATION"
02770   PRINT
02780   PRINT"              THIS COMMAND IS USED TO EVACUATE A MAJOR CITY IN ANY"
02790   PRINT"              SECTION.  THE EVACUATION PROCESS TAKES ABOUT 5 COMPUTER"
02800   PRINT"              HOURS TO COMPLETE.  IT IS NEVER REQUIRED FOR YOU TO"
02810   PRINT"              EVACUATE A CITY FOR ANY REASON.  THE COMPUTER WILL NOT"
02820   PRINT"              ALLOW THE EVACUATION OF A MAJOR CITY IN A SECTION IN"
02830   PRINT"              WHICH NO SWARMS ARE REPORTED.  NO BATTLE PHASE OPTIONS"
02840   PRINT"              CAN BE USED IN A SECTION THAT IS BEING EVACUATED UNTIL"
02850   PRINT"              IT IS COMPLETELY EVACUATED."
02860   PRINT
02870   PRINT"##############################################################"
02880   PRINT
02890   PRINT"COMMAND 5:          NUCLEAR DESTRUCTION"
02900   PRINT
02910   PRINT"              ANY CITY CAN BE DESTROYED COMPLETELY BY USING THIS COM-"
02920   PRINT"              MAND.  THE CITY DOES NOT NECESSARILY HAVE TO BE EVAC-"
02930   PRINT"              UATED TO EXECUTE THIS COMMAND.  IF THE BEES HAVE NOT"
02940   PRINT"              ENTERED THE MAJOR CITY THE COMPUTER WILL NOT ALLOW IT"
02950   PRINT"              TO BE DESTROYED USING THIS METHOD.  BEFORE USING THIS"
02960   PRINT"              COMMAND IT IS FIRST NECESSARY FOR YOU TO ENTER YOUR "
02970   PRINT"              NAME AND CODEWORD EXACTLY AS YOU DID AT THE BEGINNING."
```

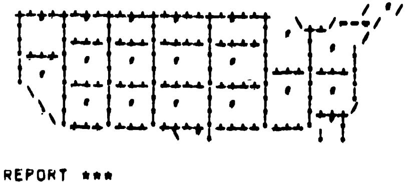
```
01000   PRINT
01010   PRINT
01020   PRINT
01030   PRINT
01040   PRINT"                    #       SWARM LARGER THAN 50,000"
01050   PRINT"                    *       SWARM BETWEEN 10,000 AND 50,000"
01060   PRINT"                    +       SWARM BETWEEN 1,000 AND 10,000"
01070   PRINT"                    .       SWARM LESS THAN 1,000 (IS REDUCED ENOUGHT TO BE"
01080   PRINT"                                                   TOTALLY DESTROYED)"
01090   PRINT
01100   PRINT"################################################################"
01110   PRINT
01120   PRINT"COMMAND 2:         ETA REPORT"
01130   PRINT
01140   PRINT"          THIS COMMAND ENABLES YOU TO FIND OUT WHEN THE BEES ARE"
01150   PRINT"          ESTIMATED TO REACH A MAJOR CITY.  TO USE THIS COMMAND"
01160   PRINT"          FOR A PARTICULAR SECTION, ENTER THIS COMMAND NUMBER(2)"
01170   PRINT"          WHEN THE COMPUTER PRINTS 'COMMAND?'.  THE COMPUTER WILL"
01180   PRINT"          THEN PRINT OUT 'SECTION?'.  ENTER THE SECTION NUMBER"
01190   PRINT"          THAT YOU WANT.  IF YOU WANT TO FIND THE ETA FOR ALL "
01200   PRINT"          MAJOR CITIES ENTER ZERO(0) AFTER THE COMPUTER PRINTS"
01210   PRINT"          'SECTION?'.  AFTER THE BEES ENTER A CITY YOU WILL"
01220   PRINT"          HAVE 6 COMPUTER HOURS TO COMBAT THEM BEFORE THEY DES-"
01230   PRINT"          STROY ALL OF THE INHABATANTS.  YOU CAN EVACUATE THE "
01240   PRINT"          CITY, HOWEVER, WHICH IS EXPLAINED AS A LATER COMMAND."
01250   PRINT"          EVERY TIME YOU EXECUTE A COMMAND, ONE COMPUTER HOUR IS"
01260   PRINT"          COUNTED.  EACH SECTION HAS ONLY ONE CITY THAT CONSID-"
01270   PRINT"          ERED MAJOR, AND THAT IS THE ONLY CITY YOU SHOULD BE"
01280   PRINT"          CONCERED WITH.  ONCE THE BEES ENTER A CITY, URBAN DEF-"
01290   PRINT"          ENSES MUST BE USED, WHICH ARE EXPLAINED LATER."
01300   PRINT
01310   PRINT"################################################################"
01320   PRINT
01330   PRINT"COMMAND 3:         BATTLE PHASE OPTIONS"
01340   PRINT
01350   PRINT"          THIS COMMAND IS WHAT YOU USE TO COMBAT THE BEES.  EACH"
01360   PRINT"          OPTION IS EXPLAINED FULLY BELOW.  HERE ARE WHAT YOUR "
01370   PRINT"          OPTIONS ARE."
01380   PRINT"                   BATTLE PHASE OPTIONS"
01390   PRINT
01400   PRINT"                   PHASE ONE:      BEE COCKTAIL"
01410   PRINT"                   PHASE TWO:      PROJECT QUEEN"
01420   PRINT"                   PHASE THREE:    PROJECT BRUSH FIRE"
01430   PRINT"                   PHASE FOUR:     PROJECT STERILE MALE"
01440   PRINT"                   PHASE FIVE:     DESTRUCTION"
01450   PRINT"                   PHASE SIX:      URBAN DEFENSES"
01460   PRINT
01470   PRINT"          EACH OF THESE COMMAND COMBATS THE BEES IN A DIFFERANT"
01480   PRINT"          WAY.  ONE COMMAND WILL WORK BETTER ON LARGER SWARMS."
01490   PRINT"          ANOTHER COMMAND WILL WORK BETTER ON SMALLER SWARMS."
01500   PRINT"          ONE COMMAND WORKS WITHIN 1 COMPUTER HOUR.  ANOTHER."
01510   PRINT"          MAY TAKE 5 HOURS.  THERE IS ALWAYS A CHANCE THAT THE"
01520   PRINT"          PHASE WILL NOT WORK AT ALL.  THE ONLY WAY TO FIND OUT "
01530   PRINT"          WHICH COMMANDS WORK ON WHICH SITUATION IS TO USE THEM."
01540   PRINT"          THE APPROXIMATE TIME IT SHOULD TAKE FOR EACH PHASE TO"
01550   PRINT"          TAKE EFFECT IS PRINTED WITH THE COMMAND.  THE TIME"
01560   PRINT"          COULD VARY BY UP TO 2 HOURS.  AS ONE SWARM IS BEING"
01570   PRINT"          COMBATED ANOTHER SWARM CAN BE BUILDING UP.  THE URBAN"
01580   PRINT"          DEFENSES ONLY WORK IN THE CITY AND SOME COULD HARM OR"
01590   PRINT"          KILL SOME OF THE POPULATION ALONG WITH THE BEES.  "
01600   PRINT"          AFTER YOU ENTER THIS COMMAND(3), THE COMPUTER WILL"
01610   PRINT"          ASK YOU FOR THE REST OF THE INFORMATION IT NEEDS."
01620   PRINT
01630   PRINT
01640   PRINT"                             EXPLANATIONS"
01650   PRINT"++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++"
01660   PRINT
01670   PRINT
01680   PRINT"*PHASE ONE:"
01690   PRINT"          BEE COCKTAIL-                             2 HOURS"
01700   PRINT"----------------------------------------------------------------"
01710   PRINT"          AMERICAN FOULBROOD"
01720   PRINT"          INSECT GROWTH REGULATOR(IGR)"
01730   PRINT"          ANTIPHEROMONE SUBSTANCE(LSD)"
01740   PRINT
01750   PRINT"     THOUSANDS OF PACKETS CONTAINING THE ABOVE SUBSTANCES IN A"
01760   PRINT"HONEY SOLUTION WILL BE DROPPED OVER INFESTED AREAS.  THESE SHOULD"
01770   PRINT"KILL SOME OF THE LARVAE, INTERFERE WITH THE DEVELOPMENT OF OTHERS,"
01780   PRINT"AND UPSET THE BEES' COMMUNICATIONS SYSTEM."
01790   PRINT
01800   PRINT"++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++"
01810   PRINT
01820   PRINT
01830   PRINT"*PHASE TWO:"
01840   PRINT"          PROJECT QUEEN-                            5 HOURS"
01850   PRINT"----------------------------------------------------------------"
01860   PRINT"          LINE-REARING QUEENS"
01870   PRINT"          NEGATIVE SURVIVAL CHARACTERISTICS"
01880   PRINT
01890   PRINT"     ARTIFICIALLY BRED QUEEN BEES WILL BE DROPPED BY HELICOPTERS"
01900   PRINT"OVER INFESTED AREAS.  THEY SHOULD MATE WITH THE AFRICANS, ENTER THE"
01910   PRINT"HIVES, AND DO BATTLE WITH THE AFRICAN QUEENS.  SOME MUST PREVAIL"
01920   PRINT"IN ORDER TO PASS ON DEFECTS TO THEIR PROGENY, AND ,IN TURN, THEIR"
01930   PRINT"QUEENS AND DRONES WILL SIMILARLY BREED AND INCREASE THE DEFECTIVE "
01940   PRINT"POPULATION UNTIL ENOUGH OF THE AFRICANS HAVE ENOUGH NEGATIVE SURV-"
01950   PRINT"IVAL CHARACTERISTICS TO MAKE THEM SELF-DESRUCTIVE."
01960   PRINT
01970   PRINT"++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++"
01980   PRINT
```

```
02980   PRINT"          IF THE POPULATION IS EVACUATED ONLY THE BEES ARE DES-"
02990   PRINT"          TROYED AND THE POPULATION WILL RETURN TO THE CITY AFTER"
03000   PRINT"          ABOUT 7 COMPUTER HOURS.  IF THE POPULATION IS NOT EVAC-"
03010   PRINT"          UATED IT IS DESTROYED WITH THE BEES AND THE ENTIRE SEC-"
03020   PRINT"          TION BECOMES UNINHABITED.  THE BEES WILL NOT ENTER IN-"
03030   PRINT"          TO A SECTION THAT IS COMPLETELY UNINHABITED, SO THAT"
03040   PRINT"          SECTION WILL BE ENTIRELY EMPTY FOR THE REMAINDER OF THE"
03050   PRINT"          GAME"
03060   PRINT
03070   PRINT"################################################################"
03080   PRINT
03090   PRINT"COMMAND 6:         CASUALTY REPORT"
03100   PRINT
03110   PRINT"          THIS COMMAND PERMITS YOU TO FIND OUT HOW MANY BEE"
03120   PRINT"          RELATED CASUALTIES THERE ARE IN THE SECTIONS.  TO"
03130   PRINT"          USE THIS COMMAND FOR ALL THE SECTIONS ENTER ZERO(0)"
03140   PRINT"          FOR THE SECTION NUMBER.  A CASUALTY REPORT IS AUTO-"
03150   PRINT"          MATICALLY ISSUED AT THE END OF A GAME UNLESS THE"
03160   PRINT"          GAME CANCEL COMMAND IS USED."
03170   PRINT
03180   PRINT"################################################################"
03190   PRINT
03200   PRINT"COMMAND 7:         COMMAND(SHORT)"
03210   PRINT
03220   PRINT"          THIS COMMAND PRINTS OUT ALL OF THE COMMANDS IN A"
03230   PRINT"          SHORT VERSION WITH NO EXPLANATIONS."
03240   PRINT
03250   PRINT"################################################################"
03260   PRINT
03270   PRINT"COMMANDS 8:        CANCEL GAME"
03280   PRINT
03290   PRINT"          THIS COMMAND CANCELS THE GAME YOU ARE PLAYING AND"
03300   PRINT"          THEN ASK YOU IF YOU WANT TO PLAY ANOTHER ONE."
03310   PRINT
03320   PRINT"################################################################"
03330   PRINT
03340   PRINT
03350   PRINT
03360   PRINT
03370   REM     *** CREATE 2 INITIAL SWARMS ***
03380   RANDOMIZE
03390   DIM E(25),S(25),A$(25),U(25),G(25),M(25),D(25),C(25),V(25),K(25),R(25)
03400   A1=INT(RND *21+1)
03410   A2=INT(RND *21+1)
03420   IF A1=A2 GOTO 3410
03430   S(A1)=INT(RND*5+2)
03440   S(A2)=INT(RND*5+2)
03450   E(A1)=INT(RND*9+6)
03460   E(A2)=INT(RND*9+6)
03470   IF I$="SHORT" THEN 8160
03480   REM     *** SUBTRACT ONE HOUR FROM ALL ETA'S ***
03490   I$="NO"
03500   FOR A=1 TO 21
03510   IF E(A)=0 THEN 3550
03520   GOTO 3540
03530   IF U(A)=0 THEN 3750
03540   IF S(A)=1 THEN 3750
03550   IF ABS(U(A))=1 THEN 3660
03560   IF ABS(E(A))=1 THEN 3600
03570   E(A)=E(A)-1
03580   C(A)=C(A)+(17-E(A))
03590   GOTO 3750
03600   IF E(A)=-1 THEN 3640
03610   PRINT"*THE BEES HAVE ARRIVED IN THE MAJOR CITY IN SECTION"A
03620   E(A)=-1
03630   U(A)=6
03640   U(A)=U(A)-1
03650   GOTO 3750
03660   IF U(A)=-1 THEN 3750
03670   PRINT"*THE BEES HAVE DESTROYED THE MAJOR CITY IN SECTION"A
03680   U(A)=-1
03690   IF V(A)<>-1 THEN 3720
03700   PRINT"BUT THE POPULATION HAS BEEN EVALUATED"
03710   GOTO 3740
03720   K(A)=1
03730   C(A)=(1000000*(RND*A+1))+C(A)
03740   GOTO 3750
03750   NEXT A
03760   REM     *** ADD ONE UNIT TO TIME ***
03770   T=T+1
03780   REM     *** CREATE NEW SWARM ***
03790   IF T/30<>INT(T/30) THEN 3880
03800   A=INT(RND*21+1)
03810   IF S(A)<>0 THEN 3790
03820   IF R(A)<>0 THEN 3790
03830   IF K(A)<>0 THEN 3790
03840   S(A)=INT(RND*5+2)
03850   E(A)=INT(RND*9+16)
03860   C(A)=INT(RND*10)
03870   PRINT"A NEW SWARM IS REPORTED IN SECTION"A
03880   REM     *** CHECK COMBATED SWARMS ***
03890   FOR A=1 TO 21
03900   IF T<>G(A) THEN 4060
03910   IF S(A)<> 1 THEN 4000
03920   S(A)=0
03930   G(A)=0
03940   E(A)=0
03950   M(A)=0
03960   K(A)=0
```

```
03970    U(A)=0
03980    PRINT"*THE SWARM IN SECTION"A"IS TOTALLY DESTROYED."
03990    GOTO 4060
04000    S(A)=S(A)-M(A)
04010    PRINT"*THE PHASE ON SECTION"A"WAS SUCCESSFUL"
04020    IF S(A)>=1 THEN 4040
04030    S(A)=1
04040    G(A)=0
04050    M(A)=0
04060    NEXT A
04070    REM     *** ADD ONE UNIT TO EACH UNCOMBATTED SWARM ***
04080    FOR A=1 TO 21
04090    IF E(A)=1 THEN 4340
04100    IF S(A)=1 THEN 4310
04110    IF S(A)=0 THEN 4340
04120    IF S(A) +1<9 THEN 4280
04130    IF A=1 THEN 4210
04140    IF S(A-1)>0 THEN 4200
04150    IF R(A-1)<>0 THEN 4340
04160    PRINT"*THE SWARM IN SECTION"A"HAS SPREAD TO SECTION"A-1
04170    S(A-1)=3
04180    E(A-1)=INT(RND *10+5)
04190    GO TO 4340
04200    IF A=21 THEN 4340
04210    IF S(A+1)>0 THEN 4340
04220    IF R(A+1)<>0 THEN 4340
04230    PRINT"*THE SWARM IN SECTION"A"HAS SPREAD TO SECTION"A+1
04240    IF R(A+1)<>0 THEN 4340
04250    S(A+1)=3
04260    E(A+1)=INT(RND *10+5)
04270    GOTO 4340
04280    S(A)=S(A)+1
04290    C(A)=C(A)+S(A)
04300    GOTO 4340
04310    IF D(A)=1 THEN 4340
04320    PRINT"*THE SWARM IN SECTION"A"IS READY TO BE DESTROYED"
04330    D(A)=1
04340    NEXT A
04350    REM     *** WINNER CHECK ***
04360    W=0
04370    FOR A=1 TO 21
04380    W=S(A)+W
04390    NEXT A
04400    IF W>=1 THEN 4470
04410    PRINT"   ******^G ALL SWARMS ARE NOW DESTROYED ^G******"
04420    PRINT"FINAL STATISTICS ON ATTACKS AND DESTRUCTION ARE NOW "
04430    PRINT"BEING COMPUTED.   ^G ^G ^G ^G ^G ^G  FINAL RESULTS FOLLOW......"
04440    A=0
04450    PRINT
04460    GOTO 7920
04470    REM     *** EVACUATION CHECK ***
04480    FOR A=1 TO 21
04490    IF V(A)<>T THEN 4590
04500    IF U(A)=-1 THEN 4560
04510    IF K(A)>0 THEN 4560
04520    PRINT"*POPULATION IN SECTION"A"IS EVACUATED"
04530    C(A)=C(A)+INT(RND*17)
04540    V(A)=+1
04550    GOTO 4590
04560    PRINT"*POPULATION IN SECTION"A"WAS DESTROYED BEFORE IT"
04570    PRINT"COULD BE EVACUATED"
04580    V(A)=0
04590    NEXT A
04600    REM     *** RETURN EVACUTATION CHECK ***
04610    FOR A=1 TO 21
04620    IF R(A)<>T THEN 4670
04630    IF R(A)<1 THEN 4670
04640    K(A)=0
04650    R(A)=0
04660    PRINT"*THE POPULATION HAS RETURNED TO THE CITY IN SECTION"A
04670    NEXT A
04680    REM     *** LOSER CHECK ***
04690    Q=0
04700    FOR A=1 TO 21
04710    Q=C(A)+Q
04720    NEXT A
04730    IF Q<75000000 THEN 4830
04740    PRINT"THE BEES HAVE DEVASTATED THE UNITED STATES^G AND"
04750    PRINT"THERE ARE NOW OVER 75 MILLION CASUALTIES.  THE BEES"
04760    PRINT"ARE NOW CONSIDERED TO BE THE VICTORS OVER MODERN TECH-"
04770    PRINT"NOLOGY.^G ^G ^G ^G ^G ^G ^G ^G ^G ^G ^G ^G ^G ^G"
04780    A=0
04790    Q=0
04800    W=0
04810    PRINT
04820    GOTO 7920
04830    REM     *** COMMAND INPUT ***
04840    PRINT
04850    PRINT"COMMAND";
04860    INPUT C
04870    PRINT"TIME:  "T+1
04880    IF C>8 THEN 4850
04890    ON C GOTO 4910,5400,5810,7040,7260,7880,8150,8420
04900    GOTO 4860
04910    REM     *** MAP PRINT OUT ***
04920    PRINT"1)          ATTACK SCAN MAP"
04930    PRINT
04940    FOR A=1 TO 21
04950    IF S(A)>6 THEN 5070
```

```
05950    IF S(A)=0 THEN 5970
05960    GOTO 5990
05970    PRINT"NO SWARM IS REPORTED IN SECTION"A
05980    GOTO 3480
05990    IF S(A)<>1 THEN 6030
06000    IF P=5 THEN 6070
06010    PRINT"THE DESTRUCTION PHASE SHOULD BE USED IN SECTION"A
06020    GOTO 3480
06030    IF E(A)<>-1 THEN 6070
06040    IF P=6 THEN 6070
06050    PRINT"URBAN DEFENSES SHOULD BE USED IN SECTION"A
06060    GOTO 3480
06070    N=RND
06080    ON P GOTO 6090,6180,6270,6330,6420,6480
06090    REM *PHASE 1*
06100    PRINT"BEE COCKTAIL: PHASE ONE, NOW BEING ATTEMPTED."
06110    IF N>.95 THEN 3480
06120    G(A)=T +INT(RND*3+1)
06130    IF S(A)>5 THEN 6160
06140    LET M(A)=S(A)-2
06150    GOTO 3480
06160    LET M(A)=S(A)-5
06170    GOTO 3480
06180    REM *PHASE 2*
06190    PRINT"PROJECT QUEEN: PHASE TWO, NOW BEING ATTEMPTED."
06200    IF N>.92 THEN 3480
06210    G(A)=T +INT(RND *3+4)
06220    IFS(A)>3 THEN 6250
06230    M(A)=1
06240    GOTO 3480
06250    M(A)=S(A)-1
06260    GOTO 3480
06270    REM *PHASE 3*
06280    PRINT"PROJECT BRUSH FIRE: PHASE THREE, NOW BEING ATTEMPTED."
06290    IF N>.96 THEN 3480
06300    G(A)=T+1
06310    M(A)=2
06320    GOTO 3480
06330    REM *PHASE 4*
06340    PRINT"PROJECT STERILE MALE: PHASE FOUR, NOW BEING ATTEMPTED."
06350    IF N>.89 THEN 3480
06360    G(A)=T+INT(RND*3+3)
06370    IF S(A)>6 THEN 6400
06380    M(A)=S(A)-1
06390    GOTO 3480
06400    M(A)=4
06410    GOTO 3480
06420    REM *PHASE 5*
06430    IF S(A)<>1 THEN 6610
06440    PRINT"DESTRUCTION: PHASE FIVE, NOW BEING ATTEMPTED."
06450    IF N>.7 THEN 3480
06460    G(A)=T+INT(RND * 3+2)
06470    GOTO 3480
06480    REM *PHASE 6*
06490    IF E(A)<>-1 THEN 6010
06500    PRINT"URBAN DEFENSE";
06510    INPUT D$
06520    C(A)=C(A)+INT(200*RND)
06530    IF RND >.8 THEN 3480
06540    IF D$="A" THEN 6630
06550    IF D$="B" THEN 6690
06560    IF D$="C" THEN 6750
06570    IF D$="D" THEN 6810
06580    IF D$="E" THEN 6870
06590    IF D$="F" THEN 6910
06600    IF D$="G" THEN 6950
06610    PRINT"INVALID URBAN DEFENSE: PLEASE START AGAIN."
06620    GOTO 3480
06630    REM *A*
06640    IF S(A)>5 THEN 6670
06650    S(A)=S(A)-1
06660    GOTO 6980
06670    S(A)=2
06680    GOTO 6980
06690    REM *B*
06700    IF S(A)>4 THEN 6730
06710    S(A)=S(A)-3
06720    GOTO 6980
06730    S(A)=S(A)-2
06740    GOTO 6980
06750    REM *C*
06760    IF S(A)>7 THEN 6790
06770    S(A)=S(A)-3
06780    GOTO 6980
06790    S(A)=1
06800    GOTO 6980
06810    REM *D*
06820    IF S(A)>3 THEN 6850
06830    S(A)=1
06840    GOTO 6980
06850    S(A)=S(A)-3
06860    GOTO 6980
06870    REM *E*
06880    IF RND>.5 THEN 6980
06890    S(A)=2
06900    GOTO 6980
06910    REM *F*
06920    IF RND >.4 THEN 6980
06930    S(A)=1
```

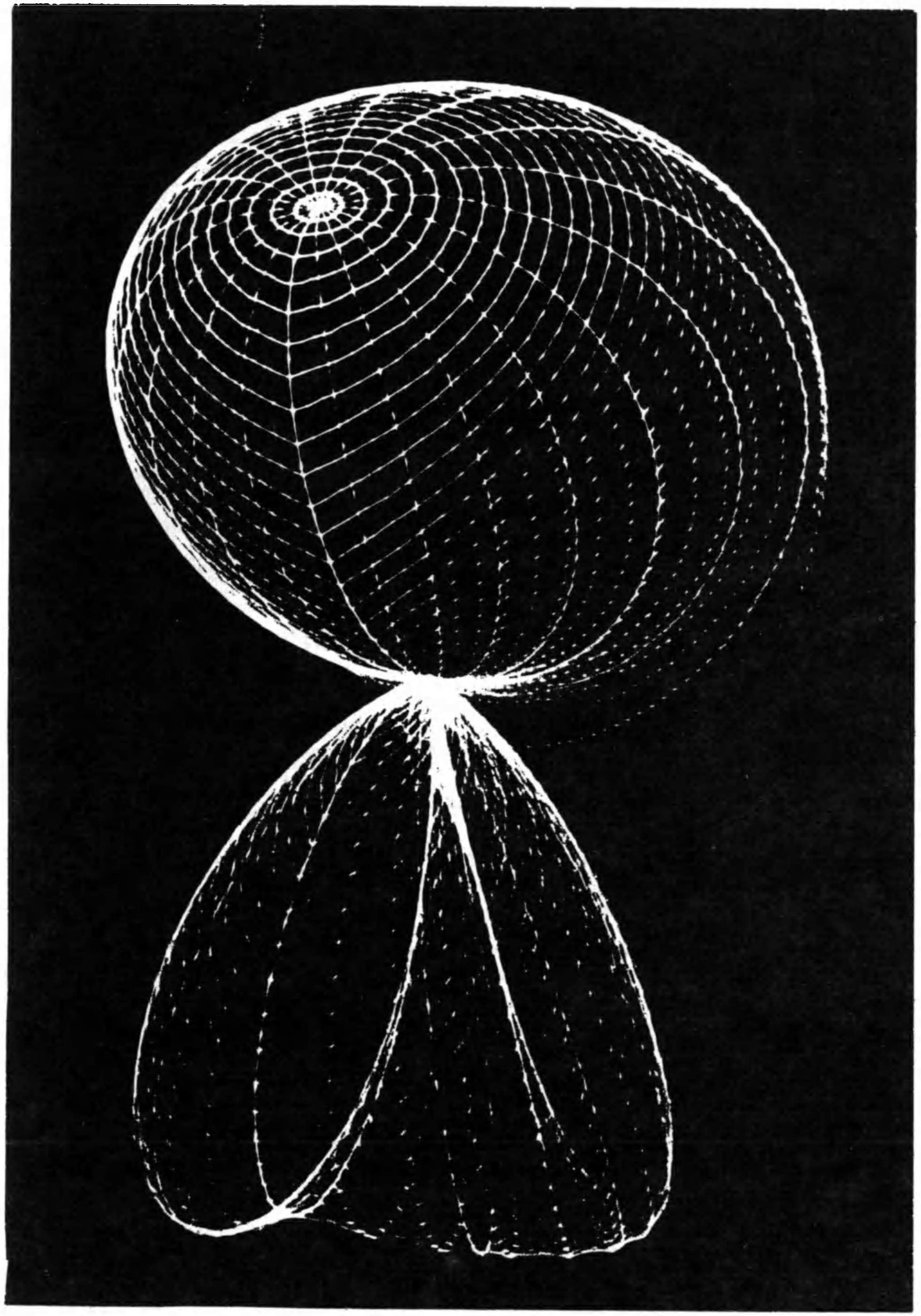
```
04960   IF S(A)>4 THEN 5050
04970   IF S(A)>1 THEN 5030
04980   IF S(A)>0 THEN 5010
04990   LET AS(A)=" "
05000   GOTO 5080
05010   LET AS(A)=","
05020   GOTO 5080
05030   LET AS(A)="+"
05040   GOTO 5080
05050   LET AS(A)="*"
05060   GOTO 5080
05070   LET AS(A)="#"
05080   NEXT A
05090   PRINT"------------------------------------------------------------"
05100   PRINT;TAB(48)"   "
05110   PRINT;TAB(20)"CANADA: NO INFORMATION      /  /"
05120   PRINT USING 5280,AS(18)
05130   PRINT;TAB(15)"|   |   |   |   |   |\   /  /"
05140   PRINT USING 5290,AS(1),AS(3),AS(6),AS(9),AS(13)
05150   PRINT USING 5300,AS(16)
05160   PRINT USING 5310,AS(19)
05170   PRINT USING 5320,AS(4),AS(7),AS(10),AS(14)
05180   PRINT USING 5330,AS(2)
05190   PRINT USING 5340,AS(17),AS(20)
05200   PRINT USING 5350,AS(5),AS(8),AS(11),AS(15)
05210   PRINT USING 5360,AS(21)
05220   PRINT;TAB(27)"\     |"TAB(42)"\ |"
05230   PRINT USING 5370,AS(12)
05240   PRINT;TAB(30)"\ |"TAB(42)"| |"
05250   PRINT;TAB(20)"MEXICO: NO INFORMATION"
05260   PRINT
05270   PRINT"------------------------------------------------------------"
05280   !
05290   !
05300   !
05310   !
05320   !
05330   !
05340   !
05350   !
05360   !
05370   !
```



```
05380   PRINT
05390   GOTO 3480
05400   REM    *** ETA REPORT ***
05410   PRINT"2)          ETA REPORT"
05420   PRINT
05430   PRINT"SECTION";
05440   INPUT A
05450   PRINT"****************************************"
05460   IF A<>0 THEN 5490
05470   LET D=-1
05480   FOR A=1 TO 21
05490   IF S(A)=1 THEN 5710
05500   IF R(A)=-1 THEN 5740
05510   IF E(A)=0 THEN 5690
05520   IF U(A)=-1 THEN 5570
05530   IF E(A)=-1 THEN 5600
05540   PRINT"THE BEES WILL ARRIVE AT THE MAJOR CITY IN"
05550   PRINT"            SECTION"A"AT"E(A)+T"HOURS."
05560   GOTO 5760
05570   PRINT"THE BEES HAVE ALREADY DESTROYED THE CITY"
05580   PRINT"IN SECTION"A"AND ARE NOW INHABITING IT."
05590   GOTO 5760
05600   PRINT"THE BEES HAVE ARRIVED IN THE CITY IN SEC-"
05610   IF V(A)<>-1 THEN 5650
05620   PRINT"TION"A"BUT THE POPULATION HAS BEEN"
05630   PRINT"                    EVACUATED"
05640   GOTO 5760
05650   PRINT"  TION"A"AND THE POPULATION OF THAT CITY"
05660   PRINT"CAN ONLY SURVIVE FOR ABOUT"U(A)"MORE HOURS."
05670   PRINT"   USE OF URBAN DEFENSES IS RECOMENDED."
05680   GOTO 5760
05690   PRINT"       NO SWARMS REPORTED IN SECTION"A
05700   GOTO 5760
05710   PRINT"THE BEES IN SECTION"A"ARE READY TO  BE"
05720   PRINT"                 DESTROYED"
05730   GOTO 5760
05740   PRINT"SECTION"A"IS A RADIOACTIVE WASTELAND THAT "
05750   PRINT"          IS COMPLETELY UNPOPULATED"
05760   PRINT"****************************************"
05770   IF D<>-1 THEN 5790
05780   NEXT A
05790   D=0
05800   GOTO 3480
05810   REM   *** BATTLE PHASE OPTIONS ***
05820   PRINT
05830   PRINT"3)          BATTLE PHASE OPTIONS"
05840   PRINT"SECTION";
05850   INPUT A
05860   PRINT"PHASE";
05870   INPUT P
05880   IF V(A)<1 THEN 5910
05890   PRINT"SECTION"A"IS BEING EVACUATED"
05900   GOTO 3480
05910   IF G(A)>0 THEN 5930
05920   GOTO 5950
05930   PRINT"THE SWARM IN SECTION"A"IS ALREADY BEING COMBATTED"
05940   GOTO 3480
```

```
06940   GOTO 6900
06950   REM *G*
06960   IF RND>.8 THEN 6980
06970   S(A)=S(A)-4
06980   PRINT"THE URBAN DEFENSE IN SECTION"A"WAS SUCESSFUL"
06990   IF S(A)>1 THEN 3480
07000   S(A)=1
07010   D(A)=1
07020   PRINT"*SWARM IN SECTION"A"IS READY TO BE DESTROYED"
07030   GOTO 3480
07040   REM     *** EVACUATION PROCEDURE ***
07050   PRINT
07060   PRINT"4)          EVACUATION PROCEDURE"
07070   PRINT"SECTION";
07080   INPUT A
07090   IF S(A)=0 THEN 7150
07100   IF V(A)>0 THEN 7170
07110   IF V(A)=-1 THEN 7190
07120   IF K(A)>0 THEN 7210
07130   IF U(A)=-1 THEN 7210
07140   GOTO 7250
07150   PRINT"COMPUTER FAILSAFE...NO SWARMS REPORTED IN SECTION"A
07160   GOTO 3480
07170   PRINT"SECTION"A"IS BEING EVACUATED ALREADY"
07180   GOTO 3480
07190   PRINT"CITY IN SECTION"A"IS ALREADY EVACUATED"
07200   GOTO 3480
07210   PRINT"POPULATION IN SECTION"A"HAS BEEN DESTROYED"
07220   GOTO 3480
07230   PRINT"EVACUATION PROCEDURE, NOW IN PROCCESS"
07240   V(A)=5+T
07250   GOTO 3480
07260   REM     *** NUCLEAR DESTRUCTION SEQUENCE ***
07270   PRINT
07280   PRINT"5)          NUCLEAR DESTRUCTION SEQUENCE"
07290   PRINT"PLEASE ENTER YOUR NAME"
07300   INPUT N1$
07310   IF N$<>N1$ THEN 7490
07320   PRINT"PLEASE ENTER YOUR CODEWORD"
07330   INPUT C1$
07340   IF C$<>C1$ THEN 7490
07350   PRINT"POSITIVE IDENTIFICATION CHECK ^G^G^G^G^G^G"
07360   PRINT"CODEWORD CHECK IS VALID ^G"
07370   PRINT"ID SEQUENCE COMPLETED"
07380   PRINT"PLEASE ENTER SECTION "
07390   INPUT A
07400   IF E(A)<>-1 THEN 7440
07410   IF S(A)=0 THEN 7470
07420   PRINT
07430   GOTO 7550
07440   PRINT"COMPUTER FAILSAFE: BEES HAVE NOT ARRIVED IN THE MAJOR "
07450   PRINT"CITY IN SECTION"A
07460   GOTO 3480
07470   PRINT"COMPUTER FAILSAFE: NO SWARM REPORTED IN SECTION"A
07480   GOTO 3480
07490   PRINT"ID SEQUENCE DEFAULT.....START AGAIN"
07500   GOTO 3480
07510   PRINT"NEUCLEAR DESTRUCTION ABORTED."
07520   GOTO 3480
07530   PRINT"BOMB IN SECTION"A"IS NOW ACTIVATED"
07540   IF K(A)=1 THEN 7600
07550   IF V(A)=-1 THEN 7600
07560   PRINT"SECTION"A"HAS NOT BEEN EVACUATED"
07570   PRINT"DO YOU WISH TO CONTINUE"
07580   INPUT K$
07590   IF K$<>"YES" THEN 7510
07600   PRINT"PUSH RETURN FOR INSTANTANIOUS DETONATION"
07610   INPUT F$
07620   PRINT"^GBOMB ^GHAS BEEN ^GDETONATED^G ^G"
07630   PRINT"SWARM IS DESTROYED^G ^G ^G"
07640   PRINT"CITY IS DESTROYED ^G ^G ^G"
07650   C(A)=C(A)+2413
07660   S(A)=0
07670   E(A)=0
07680   U(A)=0
07690   D(A)=0
07700   G(A)=0
07710   M(A)=0
07720   IF K(A)=1 THEN 7810
07730   IF V(A)<>-1 THEN 7800
07740   PRINT"THE POPULATION WILL MOVE BACK TO THE"
07750   PRINT"CITY IN SECTION"A"WHEN THE RADIATION"
07760   PRINT"LEVEL HAS DECREASED. "
07770   K(A)=2
07780   R(A)=T+7
07790   GOTO 3480
07800   C(A)=INT(2000000*(RND*A+1))+C(A)
07810   PRINT"*NO SURVIVERS ARE REPORTED IN SECTION"A
07820   PRINT"*THE BEES WILL NOT ENTER A SECITON WITH"
07830   PRINT"NO HUMAN INHABITANTS, SO SECTION"A
07840   PRINT"IS COMPLTELY LACKING LIFE OF ANY KIND"
07850   K(A)=2
07860   R(A)=-1
07870   GOTO 3480
07880   REM     *** CASUALTY REPORT ***
07890   PRINT"   6)        CASUALTY REPORT"
07900   PRINT"SECTION";
07910   INPUT A
07920   PRINT"****************************************"
```

```
07930    IF A<>0 THEN 7960
07940    LET D=-1
07950    FOR A=1 TO 21
07960    IF C(A)=0 THEN 8000
07970    IF C(A)>=1000000 THEN 8020
07980    PRINT"SECTION"A":     "C(A)"CASUALTIES REPORTED"
07990    GOTO 8030
08000    PRINT"SECTION"A":      NO BEE RELATED CASUALTIES"
08010    GOTO 8030
08020    PRINT"SECTION"A":        "C(A)/1000000"MILLION CASUALTIES"
08030    PRINT"****************************************"
08040    F=C(A)+F
08050    IF D<>-1 THEN 8120
08060    NEXT A
08070    D=0
08080    IF F<1000000 THEN 8110
08090    PRINT"TOTAL CASUALTIES REPORTED: "F/1000000"MILLION"
08100    GOTO 8120
08110    PRINT"TOTAL CASUALTIES REPORTED:"F
08120    IF W=0 THEN 8430
08130    F=0
08140    GOTO 3400
08150    REM     *** PRINT COMMANDS IN SHORT ***
08160    PRINT"************************ COMMANDS **************************"
08170    PRINT
08180    PRINT"              1) ATTACK SCAN MAP"
08190    PRINT"              2) ETA REPORT"
08200    PRINT"              3) BATTLE PHASE OPTIONS"
08210    PRINT"                     1. BEE COCKTAIL"
08220    PRINT"                     2. PROJECT QUEEN"
08230    PRINT"                     3. PROJECT BRUSH FIRE"
08240    PRINT"                     4. PROJECT STERILE MALE"
08250    PRINT"                     5. DESTRUCTION"
08260    PRINT"                     6. URBAN DEFENSES"
08270    PRINT"                               A) FLIGHT PATTERNS"
08280    PRINT"                               B) SONIC BOOM"
08290    PRINT"                               C) SUPER-SONIC BEAMS"
08300    PRINT"                               D) POLLUTION"
08310    PRINT"                               E) MYTHYL PARATHION"
08320    PRINT"                               F) FIRE WALL"
08330    PRINT"                               G) STROBE LIGHT"
08340    PRINT"              4) EVACUATION PROCEDURES"
08350    PRINT"              5) NUCLEAR DESTRUCTION"
08360    PRINT"              6) CASUALTY REPORT"
08370    PRINT"              7) COMMANDS (SHORT)"
08380    PRINT"              8) CANCEL GAME"
08390    PRINT
08400    GOTO 3400
08410    REM     *** CANCEL GAME ***
08420    PRINT"GAME CANCELLED"
08430    PRINT"DO YOU WANT TO PLAY AGAIN";
08440    INPUT X$
08450    IF X$="NO" THEN 8630
08460    PRINT"ANOTHER GAME IS NOW BEING SET UP...."
08470    MATE=ZER
08480    MATS=ZER
08490    MATU=ZER
08500    MATG=ZER
08510    MATM=ZER
08520    MATO=ZER
08530    MATC=ZER
08540    MATV=ZER
08550    MATK=ZER
08560    MATR=ZER
08570    A=W=T=P=C=D=F=0
08580    PRINT"..........READY!"
08590    PRINT
08600    PRINT
08610    PRINT
08620    GOTO 3370
08630    PRINT
08640    PRINT
08650    PRINT
08660    PRINT"    THIS GAME WAS BASED ON THE BOOK THE SWARM, BY AUTHUR HERZOG."
08670    PRINT"IT WAS CREATED AND PRODUCED BY RAND K. MILLER.  IT WAS ORIGINAL-"
08680    PRINT"LY RELEASED IN ALBUQUERQUE, N.M. ON APRIL 1,1976.     THANK YOU."
08690    END
```



Another new game from Creative Computing....

EUCHRE

by Victor Raybaud
113 Larzelere
Central Michigan University
Mt. Pleasant, MI 48858

```
CTURE.EUCHRE1
00100 REM THIS PROGRAM WRITTEN BY
00200 REM VICTOR J. RAYBAUD
00300 REM CMU SENIOR
00400 REM CPS MAJOR 2-76
00500 REM TITLE : INSTRUCTIONS TO EUCHRE FOR TWO
00600 PRINT
00700 PRINT
00800 PRINT
00900 PRINT 'THIS IS THE GAME OF TWO-HANDED EUCHRE.'
01000 PRINT 'DO YOU NEED INSTRUCTIONS? ';
01100 INPUT A$
01200 IF A$='YES' OR A$='Y' THEN 1700
01300 IF A$='NO' OR A$='N' THEN 9400
01400 PRINT 'PLEASE INPUT YES OR NO.'
01500 PRINT
01600 GOTO 1000
01700 PRINT
01800 PRINT
01900 PRINT 'THIS IS THE GAME OF TWO-HANDED EUCHRE. THE OBJECT OF'
02000 PRINT 'THE GAME IS TO GET 10 POINTS. AFTER EACH HAND ONE OF'
02100 PRINT 'US WILL GET 1,2,3, OR 4 POINTS, DEPENDING ON THE OUT-'
02200 PRINT 'COME OF EACH HAND.'
02300 PRINT
02400 PRINT 'THE DECK CONSISTS OF 24 CARDS. THE RANK FROM LOW TO'
02500 PRINT 'HIGH IS 9,10,J,Q,K,A. IF THAT SUIT IS TRUMP (EITHER'
02600 PRINT 'CLUBS, DIAMONDS, HEARTS, OR SPADES) THEN THE ORDER IS
02700 PRINT '9,10,Q,K,A,J,J. THE FIRST J IS THE LEFT BOWER (JACK '
02800 PRINT 'OF SAME COLOR BUT DIFFERENT SUIT) AND THE SECOND J IS'
02900 PRINT 'THE RIGHT BOWER (JACK OF THE SUIT CALLED TRUMP).'
03000 PRINT
03100 PRINT 'PLAY OF GAME PROCEEDS AS FOLLOWS:'
03200 PRINT
03300 PRINT '1)  EACH PERSON IS DEALT 12 CARDS IN THE FOLLOWING'
03400 PRINT '    MANNER. FOUR CARDS FACE DOWN, FOUR CARDS FACE UP'
03500 PRINT '    (ONE ON TOP OF EACH OF THE FACE DOWN CARDS) AND'
03600 PRINT '    FOUR CARDS IN YOUR HAND. LAYOUT OF THE CARDS ARE'
03700 PRINT '    IN THE FOLLOWING FORMAT:'
03800 PRINT
03900 PRINT TAB(10),'U  U  U  U'
04000 PRINT
04100 PRINT TAB(10),'*  *  *  *'
04200 PRINT TAB(10),'C  C  C  C'
04300 PRINT TAB(7),'----------------'
04400 PRINT TAB(10),'C  C  C  C'
04500 PRINT TAB(10),'*  *  *  *'
04600 PRINT
04700 PRINT TAB(10),'Y  Y  Y  Y'
04800 PRINT 'WHERE'
04900 PRINT '     * - CARDS FACE DOWN'
05000 PRINT '     C - CARDS FACE UP'
05100 PRINT '     Y - CARDS IN YOUR HAND (UNSEEN BY ME)'
05200 PRINT '     U - CARDS IN MY HAND (UNSEEN BY YOU)'
05300 PRINT
05400 PRINT 'CARDS ABOVE DOTTED LINE ARE MINE, BELOW DOTTED LINE'
05500 PRINT 'ARE YOURS. FACE DOWN CARDS ARE REVEALED ONCE THE ONE'
05600 PRINT 'ON TOP OF IT HAS BEEN PLAYED.'
05700 PRINT
05800 PRINT '2)  THE PERSON WHO DIDN''T DEAL HAS THE CHOICE OF'
05900 PRINT '    CALLING A SUIT TRUMP OR PASSING. IF HE PASSES'
06000 PRINT '    THEN THE DEALER GETS STUCK WITH CALLING TRUMP.'
06100 PRINT '    INPUT C - CLUBS, D - DIAMONDS, H - HEARTS, S -'
06200 PRINT '    SPADES, P - PASS.'
06300 PRINT
06400 PRINT '3)  THE PERSON WHO DIDN''T DEAL LEADS WITH FIRST PLAY.'
06500 PRINT '    FOLLOW PLAY WITH SAME SUIT. IF YOU DON''T HAVE'
06600 PRINT '    THAT SUIT THEN PLAY ANYTHING, TRUMP TAKES ALL,'
06700 PRINT '    EXCEPT HIGHER TRUMP.'
06800 PRINT
06900 PRINT '4)  THE WINNER OF PLAY LEADS NEXT PLAY. CONTINUE PLAY'
07000 PRINT '    OF HAND UNTIL ALL CARDS ARE EXHAUSTED.'
07100 PRINT
07200 PRINT '5)  INPUT CARD TO BE PLAYED BY THE FOLLOWING FORMAT:'
07300 PRINT '      CARD RANK , SUIT'
07400 PRINT '    RANK - 9,10,J,Q,K,A'
07500 PRINT '    SUIT - C,D,H,S'
07600 PRINT '    EXAMPLE:   Q,S      QUEEN OF SPADES'
07700 PRINT
07800 PRINT '6)  WHOEVER CALLS TRUMP HAS AN OBLIGATION TO WIN AT'
07900 PRINT '    LEAST SEVEN TRICKS. IF NOT THEN HE IS EUCHRED.'
08000 PRINT
08100 PRINT '7)  POINTS ARE ASSIGNED AS FOLLOWS:'
08200 PRINT '    IF YOU CALL TRUMP YOU GET'
08300 PRINT '     1 POINT  - IF YOU WIN AT LEAST 7 TRICKS'
08400 PRINT '     2 POINTS - IF YOU WIN 9 OR 10 TRICKS'
08500 PRINT '     3 POINTS - IF YOU WIN 11 TRICKS'
08600 PRINT '     4 POINTS - IF YOU WIN ALL 12 TRICKS'
08700 PRINT '    IF YOU DIDN''T CALL TRUMP YOU GET'
08800 PRINT '     2 POINTS - IF YOU WIN AT LEAST 6 TRICKS (EUCHRE)'
08900 PRINT '     4 POINTS - IF YOU WIN ALL 12 TRICKS'
09000 PRINT
09100 PRINT '8)  DEALS ALTERNATE AT THE END OF EACH HAND.'
09200 PRINT
09300 PRINT '9)  MOST IMPORTANT - NO CHEATING !!!!!!!'
09400 PRINT
09500 PRINT
09600 CHAIN EUCHRE2
09700 END
```

Language: BASIC (Univac 1106)

Description: This game pits the user against the computer in a card game called Euchre. See the program for instructions. (Note: if you haven't played or even heard of Euchre before, they may be a little hard to understand.)

EUCHRE uses several functions which may not be familiar to you. They are:

INP-	Takes the INteger Part of a number. Same as INT in most BASICs.
MOD-	MOD(X,Y) returns the value of X Mod Y. For example, 154 Mod 100 is 54, 299 Mod 100 is 99, 12 Mod 100 is 12, and -20 Mod 100 is 80.
CAT!-	Concatenates two strings ("adds" them).

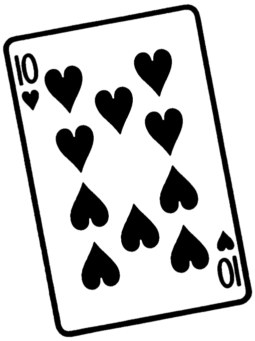
Notice also that Univac 1106 Basic permits any statement to follow an IF...THEN. Have "fun" converting this to your BASIC if it doesn't have this feature! Also remember that the strings are string vectors, so Z$(I) refers to a whole group of characters (and not the Ith character in a string, as in HP BASIC).

EUCHRE1 (to the left) contains instructions for playing the game.

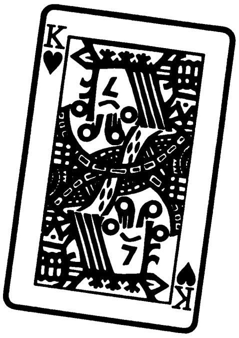
EUCHRE2 (below and following two pages) is the game itself. The third page over contains part of a sample run.

```
      EUCHRE2
00100 REM THIS PROGRAM WRITTEN BY
00200 REM VICTOR J. RAYBAUD
00300 REM CMU-SENIOR
00400 REM CPS MAJOR 2-76
00500 REM TITLE : EUCHRE FOR TWO
00600 DIM B(24),Z$(24),D(4),G(24),N(12),P$(12)
00700 DIM K$(12),L$(12),Q(24),R$(24),S(12),U$(12)
00800 DIM V(4),W(4),E(8),F(8)
00900 RANDOMIZE
01000 FOR I=1 TO 24
01100 READ B(I),Z$(I)
01200 NEXT I
01300 MAT G=ZER(24)
01400 LET P3=P4=T1=T2=0
01500 PRINT
01600 PRINT 'WOULD YOU LIKE TO DEAL FIRST? ';
01700 INPUT A$
01800 IF A$='YES' OR A$='Y' THEN 2200
01900 IF A$='NO' OR A$='N' THEN 2500
02000 PRINT 'PLEASE INPUT YES OR NO.'
02100 GOTO 1500
02200 LET W1=1
02300 LET W2=2
02400 GOTO 2700
02500 LET W1=2
02600 LET W2=1
02700 FOR I=1 TO 24
```

```
02800 LET X1=INP(100*(RND(E2)))
02900 LET Y=MOD(X1,24)+1
03000 IF I=1 THEN 3800
03100 FOR J=1 TO I-1
03200 IF G(J)=Y THEN 3500
03300 NEXT J
03400 GOTO 3800
03500 LET Y=Y+7
03600 IF Y>24 THEN Y=Y-24
03700 GOTO 3100
03800 LET R$(I)=Z$(Y)
03900 LET Q(I)=B(Y)
04000 LET G(I)=Y
04100 NEXT I
04200 LET J=1
04300 FOR I=1 TO 23 STEP 2
04400 LET P$(J)=R$(I)
04500 LET N(J)=Q(I)
04600 LET U$(J)=R$(I+1)
04700 LET S(J)=Q(I+1)
04800 LET J=J+1
04900 NEXT I
05000 FOR I=1 TO 4
05100 LET K$(I)=P$(I)
05200 LET K$(I+4)=P$(I+4)
05300 LET L$(I)=U$(I)
05400 LET L$(I+4)=' U '
05500 NEXT I
05600 FOR I=9 TO 12
05700 LET K$(I)=L$(I)='***'
05800 NEXT I
05900 PRINT
06000 PRINT
06100 IF W1=1 THEN 6400
06200 PRINT 'I DEALT THE FOLLOWING CARDS:'
06300 GOTO 6500
06400 PRINT 'YOU DEALT THE FOLLOWING CARDS:'
06500 GOSUB 47900
06600 IF W1=1 THEN 9200
06700 PRINT
06800 PRINT 'WHAT''S TRUMP? ';
06900 INPUT T$
07000 IF T$='C' THEN 7700
07100 IF T$='D' THEN 7900
07200 IF T$='S' THEN 8100
07300 IF T$='H' THEN 8300
07400 IF T$='P' THEN 8500
07500 PRINT 'PLEASE INPUT C,D,S,H, OR P.'
07600 GOTO 6700
07700 PRINT 'CLUBS ARE TRUMP'
07800 GOTO 9000
07900 PRINT 'DIAMONDS ARE TRUMP'
08000 GOTO 9000
08100 PRINT 'SPADES ARE TRUMP'
08200 GOTO 9000
08300 PRINT 'HEARTS ARE TRUMP'
08400 GOTO 9000
08500 IF W1=1 THEN 8800
08600 PRINT 'YOU PASSED, SO I SAY ';
08700 GOTO 9400
08800 PRINT 'SORRY, YOU GET STUCK CALLING TRUMP!!!'
08900 GOTO 6700
09000 LET W3=1
09100 GOTO 17100
09200 PRINT
09300 PRINT 'YOU DEALT SO I SAY ';
09400 MAT V=ZER(4)
09500 FOR I=1 TO 4
09600 FOR J=1 TO 12
09700 LET I1=INP(N(J)/100)
09800 IF I1=I THEN V(I)=V(I)+N(J)
09900 NEXT J
10000 NEXT I
10100 FOR I=1 TO 12
10200 LET A1=MOD(N(I),100)
10300 LET I1=INP(N(I)/100)
10400 IF A1=11 THEN 10600
10500 GOTO 11800
10600 ON I1 GOTO 10700,11000,11300,11600
10700 LET V(1)=V(1)+5
10800 LET V(2)=V(2)+215
10900 GOTO 11800
11000 LET V(2)=V(2)+5
11100 LET V(1)=V(1)+115
11200 GOTO 11800
11300 LET V(3)=V(3)+5
11400 LET V(4)=V(4)+415
11500 GOTO 11800
11600 LET V(4)=V(4)+5
11700 LET V(3)=V(3)+315
11800 NEXT I
11900 MAT W=ZER(4)
12000 FOR I=1 TO 4
12100 FOR J=1 TO 12
12200 LET I2=INP(S(J)/100)
12300 IF I2=I THEN W(I)=W(I)+S(J)
12400 NEXT J
12500 NEXT I
12600 FOR J=1 TO 12
12700 LET A2=MOD(S(J),100)
12800 LET I2=INP(S(J)/100)
12900 IF A2=11 THEN 13100
13000 GOTO 14300
13100 ON I2 GOTO 13200,13500,13800,14100
13200 LET W(1)=W(1)+5
13300 LET W(2)=W(2)+215
13400 GOTO 14300
13500 LET W(2)=W(2)+5
13600 LET W(1)=W(1)+115
13700 GOTO 14300
13800 LET W(3)=W(3)+5
13900 LET W(4)=W(4)+415
14000 GOTO 14300
14100 LET W(4)=W(4)+5
14200 LET W(3)=W(3)+315
14300 NEXT J
14400 FOR I=1 TO 4
14500 LET D(I)=W(I)-V(I)
14600 NEXT I
14700 LET I=1
14800 FOR J=2 TO 4
14900 IF D(I)>D(J) THEN 15100
15000 LET I=J
15100 NEXT J
15200 IF D(I)<300 THEN 16800
15300 ON I GOTO 15400,15700,16000,16300,16600
15400 PRINT 'CLUBS ARE TRUMP'
15500 LET T$='C'
15600 GOTO 17000
15700 PRINT 'SPADES ARE TRUMP'
15800 LET T$='S'
15900 GOTO 17000
16000 PRINT 'DIAMONDS ARE TRUMP'
16100 LET T$='D'
16200 GOTO 17000
16300 PRINT 'HEARTS ARE TRUMP'
16400 LET T$='H'
```

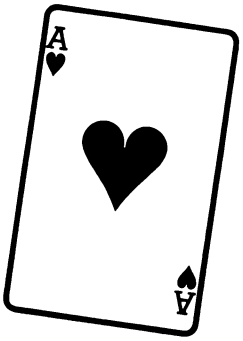


```
16500 GOTO 17000
16600 PRINT 'PASS'
16700 GOTO 6700
16800 IF W1=1 THEN I=5 ELSE PRINT 'THANKS A LOT!!!'
16900 GOTO 15300
17000 LET W3=2
17100 IF T$='C' THEN 17700
17200 IF T$='D' THEN 18000
17300 IF T$='H' THEN 18300
17400 LET B1=1
17500 LET B2=2
17600 GOTO 18500
17700 LET B1=2
17800 LET B2=1
17900 GOTO 18500
18000 LET B1=4
18100 LET B2=3
18200 GOTO 18500
18300 LET B1=3
18400 LET B2=4
18500 LET C=B1*100+11
18600 FOR I=1 TO 12
18700 IF N(I)=C THEN N(I)=B2*100+15
18800 IF S(I)=C THEN S(I)=B2*100+15
18900 NEXT I
19000 LET C=B2*100+11
19100 FOR I=1 TO 12
19200 IF N(I)=C THEN N(I)=B2*100+16
19300 IF S(I)=C THEN S(I)=B2*100+16
19400 NEXT I
19500 IF W2=2 THEN 24900
19600 PRINT
19700 PRINT 'PLAY A CARD ';
19800 INPUT R3$,S3$
19900 GOSUB 51400
20000 LET C3$=CAT$(R3$,S3$)
20100 FOR I=1 TO 12
20200 LET B3$=TRM$(P$(I))
20300 IF B3$=C3$ THEN 20500
20400 NEXT I
20500 LET V3=N(I)
20600 LET S5=INP(V3/100)
20700 LET A1=MOD(V3,100)
20800 LET J=1
20900 FOR I=1 TO 8
21000 LET F1=INP(S(I)/100)
21100 IF F1<>S5 THEN 21400
21200 LET E(J)=S(I)
21300 LET J=J+1
21400 NEXT I
21500 IF J=1 THEN 22500
21600 IF J=2 THEN V4=E(J-1) ELSE 21800
21700 GOTO 31000
21800 FOR I=1 TO J-1
21900 LET M=MOD(E(I),100)-A1
22000 IF M<0 THEN 22300
22100 LET V4=E(I)
22200 GOTO 31000
22300 NEXT I
22400 GOTO 24100
22500 FOR I=1 TO 8
22600 LET F1=INP(S(I)/100)
22700 IF F1<>B2 THEN 23000
22800 LET E(J)=S(I)
22900 LET J=J+1
23000 NEXT I
23100 IF J<>1 THEN 24000
23200 FOR I=1 TO 8
23300 LET F2=MOD(S(I),100)
23400 IF F2=J+8 THEN 23800
23500 NEXT I
23600 LET J=J+1
23700 GOTO 23200
23800 LET V4=S(I)
23900 GOTO 31000
24000 IF J=2 THEN 24700
24100 LET M=1
24200 FOR I=2 TO J-1
24300 IF E(M)>E(I) THEN M=I
24400 NEXT I
24500 LET V4=E(M)
24600 GOTO 31000
24700 LET V4=E(J-1)
24800 GOTO 31000
24900 LET H=0
25000 FOR J=1 TO 4
25100 IF J=B2 THEN 29700
25200 LET M=1
25300 FOR I=1TO 8
25400 LET S5=INP(N(I)/100)
25500 IF J<>S5 THEN 25800
25600 LET F(M)=N(I)
25700 LET M=M+1
25800 NEXT I
25900 IF M=1 THEN 27600
26000 IF M<>2 THEN 26300
26100 LET V3=F(1)
26200 GOTO 26800
26300 LET Z1=1
26400 FOR I=2 TO M-1
26500 IF F(Z1)<F(I) THEN Z1=I
26600 NEXT I
26700 LET V3=F(Z1)
26800 FOR I=1 TO 8
26900 LET S6=INP(S(I)/100)
27000 IF S6<>J THEN 27200
27100 IF S(I)>V3 THEN 27400
27200 NEXT I
27300 GOTO 29700
27400 LET V4=S(I)
27500 GOTO 31000
27600 FOR I=1 TO 8
27700 LET S5=INP(N(I)/100)
27800 IF S5=B2 THEN 29700
27900 NEXT I
28000 LET M=1
28100 FOR I=1 TO 8
28200 LET S6=INP(S(I)/100)
28300 IF S6<>J THEN 28600
28400 LET E(M)=S(I)
28500 LET M=M+1
28600 NEXT I
28700 IF M=1 THEN 29700
28800 IF M<>2 THEN 29100
28900 LET V4=E(1)
29000 GOTO 31000
29100 LET J=1
29200 FOR I=1 TO M-1
29300 IF E(J)>E(I) THEN J=I
29400 NEXT I
29500 LET V4=E(J)
29600 GOTO 31000
29700 IF H=1 THEN 30200
29800 NEXT J
29900 LET H=1
30000 LET J=B2
30100 GOTO 25200
30200 LET H=9
```

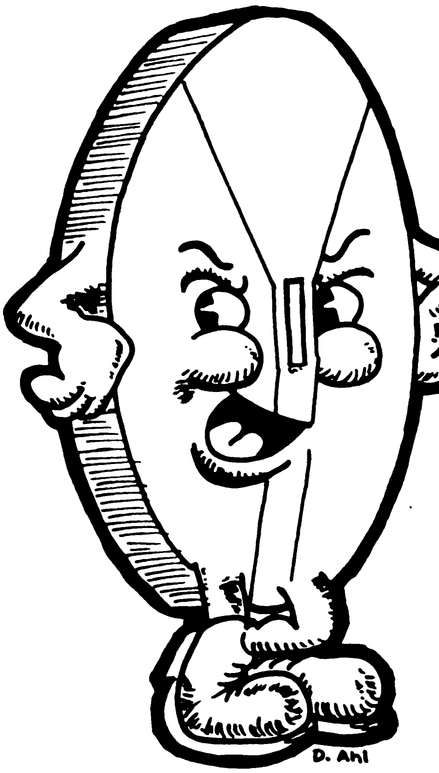


```
30300 FOR I=1 TO 8
30400 LET A2=MOD(S(I),100)
30500 IF A2=H THEN 30900
30600 NEXT I
30700 LET H=H+1
30800 GOTO 30300
30900 LET V4=S(I)
31000 LET S6=INP(V4/100)
31100 LET A2=MOD(V4,100)-8
31200 ON S6 GOTO 31300,31600,31900,32200
31300 LET S4$='C'
31400 IF A2=7 THEN S4$='S'
31500 GOTO 32400
31600 LET S4$='S'
31700 IF A2=7 THEN S4$='C'
31800 GOTO 32400
31900 LET S4$='D'
32000 IF A2=7 THEN S4$='H'
32100 GOTO 32400
32200 LET S4$='H'
32300 IF A2=7 THEN S4$='D'
32400 ON A2 GOTO 32500,32700,32900,33100,33300,
32500 LET R4$='9'
32600 GOTO 33600
32700 LET R4$='10'                33500,32900,32900
32800 GOTO 33600
32900 LET R4$='J'
33000 GOTO 33600
33100 LET R4$='Q'
33200 GOTO 33600
33300 LET R4$='K'
33400 GOTO 33600
33500 LET R4$='A'
33600 LET C4$=CAT$(R4$,S4$)
33700 IF W2=1 THEN 34400
33800 PRINT
33900 PRINT 'I PLAY ';C4$
34000 PRINT 'PLAY A CARD';
34100 INPUT R3$,S3$
34200 GOSUB 51400
34300 GOTO 34500
34400 PRINT 'I PLAY ';C4$
34500 LET C3$=CAT$(R3$,S3$)
34600 FOR I=1 TO 12
34700 LET B3$=TRM$(P$(I))
34800 IF B3$=C3$ THEN 35000
34900 NEXT I
35000 LET V3=N(I)
35100 LET S5=INP(V3/100)
35200 LET S6=INP(V4/100)
35300 IF W2=1 THEN 36500
35400 IF S5=S6 THEN 36000
35500 IF S5=B2 THEN 36100
35600 LET P4=P4+1
35700 LET W2=2
35800 PRINT 'MY TRICK'
35900 GOTO 36800
36000 IF V3<V4 THEN 35600
36100 LET P3=P3+1
36200 LET W2=1
36300 PRINT 'YOUR TRICK'
36400 GOTO 36800
36500 IF S5=S6 THEN 36000
36600 IF S6=B2 THEN 35600
36700 GOTO 36100
36800 FOR I=5 TO 8
36900 LET B3$=TRM$(P$(I))
37000 IF B3$=C3$ THEN 37300
37100 NEXT I
37200 GOTO 37600
37300 LET K$(I)=P$(I)='
37400 LET N(I)=0
37500 GOTO 38700
37600 FOR I=1 TO 4
37700 LET B3$=TRM$(P$(I))
37800 IF B3$=C3$ THEN 38100
37900 NEXT I
38000 GOTO 38700
38100 LET K$(I)=P$(I+8)
38200 LET K$(I+8)='    '
38300 LET P$(I)=P$(I+8)
38400 LET P$(I+8)='    '
38500 LET N(I)=N(I+8)
38600 LET N(I+8)=0
38700 LET C4$=CAT$(R4$,S4$)
38800 FOR I=5 TO 8
38900 LET B3$=TRM$(U$(I))
39000 IF B3$=C4$ THEN 39300
39100 NEXT I
39200 GOTO 39600
39300 LET L$(I)=U$(I)='
39400 LET S(I)=0
39500 GOTO 40700
39600 FOR I=1 TO 4
39700 LET B3$=TRM$(U$(I))
39800 IF B3$=C4$ THEN 40100
39900 NEXT I
40000 GOTO 40700
40100 LET L$(I)=U$(I+8)
40200 LET L$(I+8)='    '
40300 LET U$(I)=U$(I+8)
40400 LET U$(I+8)='    '
40500 LET S(I)=S(I+8)
40600 LET S(I+8)=0
40700 LET S7=P3+P4
40800 IF S7=12 THEN 41200
40900 PRINT 'TRICKS: ME';P4;'YOU';P3
41000 GOSUB 47900
41100 GOTO 19500
41200 PRINT 'TRICKS AT END OF HAND: ME';P4;'YOU';P3
41300 IF W3=1 THEN 43800
41400 IF P4=0 THEN 42000
41500 IF P4<7 THEN 42300
41600 IF P4=7 OR P4=8 THEN 42600
41700 IF P4=9 OR P4=10 THEN 42900
41800 IF P4=11 THEN 43200
41900 IF P4=12 THEN 43500
42000 PRINT 'WHAT LUCK, YOU GET 4 POINTS'
42100 LET T1=T1+4
42200 GOTO 45300
42300 PRINT 'EUCHRE, YOU GET 2 POINTS'
42400 LET T1=T1+2
42500 GOTO 45300
42600 PRINT 'I GOT MY 7 TRICKS'
42700 LET T2=T2+1
42800 GOTO 45300
42900 PRINT 'I GOT';P4;'TRICKS, I GET 2 POINTS'
43000 LET T2=T2+2
43100 GOTO 45300
43200 PRINT 'I GOT 11 TRICKS, I GET 3 POINTS'
43300 LET T2=T2+3
43400 GOTO 45300
43500 PRINT 'WHAT SKILL, I GET 4 POINTS'
43600 LET T2=T2+4
43700 GOTO 45300
43800 IF P3=0 THEN 43500
43900 IF P3<7 THEN 44400
```

```
44000 IF P3=7 OR P3=8 THEN 44600
44100 IF P3=9 OR P3=10 THEN 44900
44200 IF P3=11 THEN 45100
44300 IF P3=12 THEN 42000
44400 PRINT 'EUCHRE, I GET 2 POINTS'
44500 GOTO 43000
44600 PRINT 'YOU GOT YOUR 7 TRICKS'
44700 LET T1=T1+1
44800 GOTO 45300
44900 PRINT 'YOU GOT';P3;'TRICKS, YOU GET 2 POINTS'
45000 GOTO 42400
45100 PRINT 'YOU GOT 11 TRICKS, YOU GET 3 POINTS'
45200 LET T1=T1+3
45300 IF T1>=10 THEN 46900
45400 IF T2>=10 THEN 47100
45500 PRINT 'ME ';T2;' YOU ';T1
45600 IF W1=1 THEN 46200
45700 LET W1=1
45800 LET W2=2
45900 PRINT
46000 PRINT 'I DEALT LAST HAND, YOUR TURN TO DEAL'
46100 GOTO 46600
46200 LET W1=2
46300 LET W2=1
46400 PRINT
46500 PRINT 'YOU DEALT LAST HAND, MY TURN TO DEAL'
46600 MAT G=ZER(24)
46700 LET P3=P4=0
46800 GOTO 2700
46900 PRINT 'CONGRATULATIONS, YOU WIN';T1;'TO';T2
47000 GOTO 47200
47100 PRINT 'I WON THIS GAME';T2;'TO';T1
47200 PRINT
47300 PRINT 'WOULD YOU LIKE TO PLAY ANOTHER GAME?';
47400 INPUT A$
47500 IF A$='YES' OR A$='Y' THEN 1300
47600 IF A$='NO' OR A$='N' THEN 55900
47700 PRINT 'PLEASE INPUT YES OR NO.'
47800 GOTO 47300
47900 FOR I=5 TO 7
48000 IF L$(I)<>'    ' THEN 48700
48100 LET L$(I)=L$(I+1)
48200 LET L$(I+1)='    '
48300 LET U$(I)=U$(I+1)
48400 LET U$(I+1)='    '
48500 LET S(I)=S(I+1)
48600 LET S(I+1)=0
48700 NEXT I
48800 FOR I=5 TO 7
48900 IF K$(I)<>'    ' THEN 49600
49000 LET K$(I)=K$(I+1)
49100 LET K$(I+1)='    '
49200 LET P$(I)=P$(I+1)
49300 LET P$(I+1)='    '
49400 LET N(I)=N(I+1)
49500 LET N(I+1)=0
49600 NEXT I
49700 PRINT
49800 PRINT 'LAYOUT OF CARDS'
49900 PRINT
50000 PRINT L$(5);'    ';L$(6);'    ';L$(7);'    ';L$(8)
50100 PRINT
50200 PRINT '   ';'    ';L$(9);'    ';L$(10);'    ';L$(11);'    ';L$(12)
50300 PRINT '   ';'    ';L$(1);'    ';L$(2);'    ';L$(3);'    ';L$(4)
50400 PRINT '  ----------------------------'
50500 PRINT '   ';'    ';K$(1);'    ';K$(2);'    ';K$(3);'    ';K$(4)
50600 PRINT '   ';'    ';K$(9);'    ';K$(10);'    ';K$(11);'    ';K$(12)
50700 PRINT
50800 IF K$(5)='    ' THEN 51200
50900 PRINT 'CARDS IN YOUR HAND:'
51000 PRINT K$(5);'    ';K$(6);'    ';K$(7);'    ';K$(8)
51100 GOTO 51300
51200 PRINT 'NO CARDS IN YOUR HAND'
51300 RETURN
51400 IF R3$='9' THEN 52400
51500 IF R3$='10' THEN 52400
51600 IF R3$='J' THEN 52400
51700 IF R3$='Q' THEN 52400
51800 IF R3$='K' THEN 52400
51900 IF R3$='A' THEN 52400
52000 PRINT 'FORMAT OF INPUT IS INCORRECT'
52100 PRINT 'PLAY A CARD ';
52200 INPUT R3$,S3$
52300 GOTO 51400
52400 IF S3$='S' THEN 52900
52500 IF S3$='D' THEN 52900
52600 IF S3$='H' THEN 52900
52700 IF S3$='C' THEN 52900
52800 GOTO 52000
52900 LET C3$=CAT$(R3$,S3$)
53000 FOR I=1 TO 8
53100 LET B3$=TRM$(P$(I))
53200 IF C3$=B3$ THEN 53700
53300 NEXT I
53400 PRINT 'RULE NO. 9 ---- NO CHEATING!!!!!'
53500 PRINT 'YOU DON''T HAVE THAT CARD!'
53600 GOTO 52100
53700 IF W2=1 THEN 55200
53800 LET S6=INP(V4/100)
53900 LET S5=INP(N(I)/100)
54000 IF S5=S6 THEN 55200
54100 FOR I=1 TO 8
54200 LET S5=INP(N(I)/100)
54300 IF S5<>S6 THEN 55100
54400 PRINT 'RULE NO. 9 ----- NO CHEATING!!!!!'
54500 PRINT 'RULE NO. 3 ----- FOLLOW PLAY WITH SAME SUIT.'
54600 PRINT 'YOU CAN FOLLOW SUIT!!!'
54700 PRINT 'PLAY A CARD ';
54800 INPUT R3$,S3$
54900 GOSUB 51400
55000 GOTO 53700
55100 NEXT I
55200 RETURN
55300 DATA 109,'9C ',110,'10C',111,'JC ',112,'QC '
55400 DATA 113,'KC ',114,'AC ',209,'9S ',210,'10S'
55500 DATA 211,'JS ',212,'QS ',213,'KS ',214,'AS '
55600 DATA 309,'9D ',310,'10D',311,'JD ',312,'QD '
55700 DATA 313,'KD ',314,'AD ',409,'9H ',410,'10H'
55800 DATA 411,'JH ',412,'QH ',413,'KH ',414,'AH '
55900 PRINT
56000 PRINT 'THANKS FOR PLAYING, COME AGAIN!'
56100 END
```



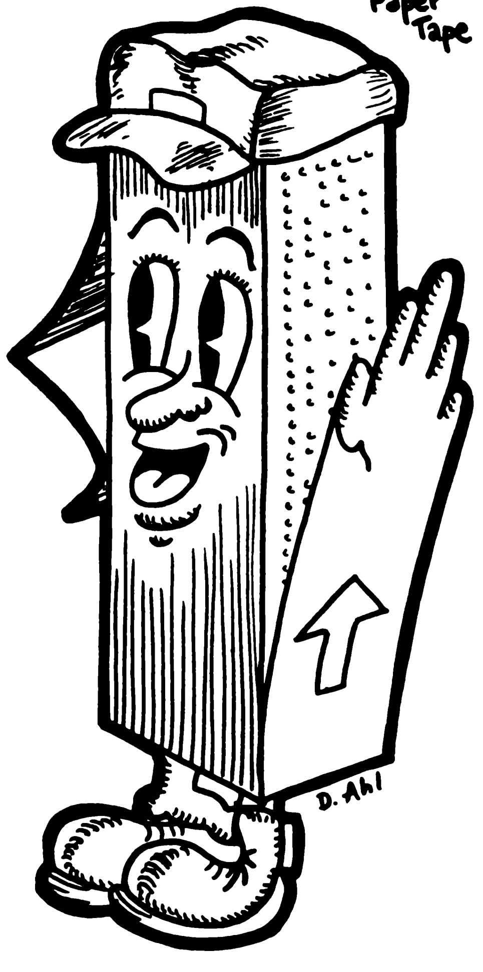
```
WOULD YOU LIKE TO DEAL FIRST? ? >NO

I DEALT THE FOLLOWING CARDS:

LAYOUT OF CARDS

 U     U     U     U

      ***   ***   ***   ***
      9S    QS    KH    KC
  -----------------------------
      9C    10C   10H   JD
      ***   ***   ***   ***

CARDS IN YOUR HAND:
10S   QH    9H    JS

WHAT'S TRUMP? ? >P
YOU PASSED, SO I SAY DIAMONDS ARE TRUMP

PLAY A CARD ? >10,H
I PLAY KH
MY TRICK
TRICKS: ME 1 YOU 0

LAYOUT OF CARDS

 U     U     U     U

      ***   ***         ***
      9S    QS    JH    KC
  -----------------------------
      9C    10C   AH    JD
      ***   ***         ***

CARDS IN YOUR HAND:
10S   QH    9H    JS

I PLAY KC
PLAY A CARD? >10,C
MY TRICK
TRICKS: ME 2 YOU 0

LAYOUT OF CARDS

 U     U     U     U

      ***   ***
      9S    QS    JH    KS
  -----------------------------
      9C    KD    AH    JD
      ***               ***

CARDS IN YOUR HAND:
10S   QH    9H    JS

I PLAY QS
PLAY A CARD? >10,S
MY TRICK
TRICKS: ME 3 YOU 0

LAYOUT OF CARDS

 U     U     U     U

      ***
      9S    10D   JH    KS
  -----------------------------
      9C    KD    AH    JD
      ***               ***

CARDS IN YOUR HAND:
QH    9H    JS

I PLAY KS
PLAY A CARD? >A,S
RULE NO. 9 ---- NO CHEATING!!!!!
YOU DON'T HAVE THAT CARD!
PLAY A CARD ? >J,D
RULE NO. 9 ----- NO CHEATING!!!!!
RULE NO. 3 ----- FOLLOW PLAY WITH SAME SUIT.
YOU CAN FOLLOW SUIT!!!
PLAY A CARD ? >J,S
MY TRICK
TRICKS: ME 4 YOU 0

LAYOUT OF CARDS

 U     U     U     U

      ***
      9S    10D   JH
  -----------------------------
      9C    KD    AH    JD
      ***               ***

CARDS IN YOUR HAND:
QH    9H

I PLAY 9S
PLAY A CARD? >K,D
YOUR TRICK
TRICKS: ME 4 YOU 1

LAYOUT OF CARDS

 U     U     U     U


      JC    10D   JH
  -----------------------------
      9C          AH    JD
      ***               ***

CARDS IN YOUR HAND:
QH    9H

PLAY A CARD ? >J,D
I PLAY 9D
YOUR TRICK
TRICKS: ME 4 YOU 2

LAYOUT OF CARDS

 U     U     U

      JC    10D   JH
  -----------------------------
      9C          AH    QC
      ***

CARDS IN YOUR HAND:
QH    9H

PLAY A CARD ? >Q,C
I PLAY JC
YOUR TRICK
TRICKS: ME 4 YOU 3

LAYOUT OF CARDS

 U     U     U


            10D   JH
  -----------------------------
      9C          AH
      ***

CARDS IN YOUR HAND:
QH    9H

PLAY A CARD ? >Q,H
I PLAY 10D
MY TRICK
TRICKS: ME 5 YOU 3

LAYOUT OF CARDS

 U     U     U


                  JH
  -----------------------------
      9C          AH
      ***

CARDS IN YOUR HAND:
9H

I PLAY AS
PLAY A CARD? >9,C
MY TRICK
TRICKS: ME 6 YOU 3

LAYOUT OF CARDS

 U     U


                  JH
  -----------------------------
      AC          AH


CARDS IN YOUR HAND:
9H

I PLAY QD
PLAY A CARD? >9,H
MY TRICK
TRICKS: ME 7 YOU 3

LAYOUT OF CARDS

 U


                  JH
  -----------------------------
      AC          AH


NO CARDS IN YOUR HAND

I PLAY AD
PLAY A CARD? >A,H
MY TRICK
TRICKS: ME 8 YOU 3
```



MORE COMPUTER CRITTERS!!

← Devilish Donald Discpack

← Patrick Paper Tape



Another new game from Creative Computing . . . .

DAYTONA 500

OR

A SYNCHRONOUS SIMULATION MODEL OF VEHICULAR COMPETITION WITH STOCHASTIC DECREMENTATION OF THE COMPETITIVE UNIVERSE

Geoffrey Churchill
Georgia State University

No, I'm not really *that* pompous, but won't it look fantastic on my resume? Needless to say, I'm betting the Dean doesn't read this. If I didn't need a nice-looking article on the list, I'd just call it the Daytona 500 Game.

Anyway, if you're a frustrated hotshoe driver, this game is for you. It is set up for six competitors on a 2½-mile track, but if you're overloaded with core and cpu time it would be easy to modify for a full field of cars. N-person computer games have been hard for me to find, which is why I wrote it; I've had fun with it.

To get an edge in figuring your strategy, you should be aware of the expressions on lines 380 to 430 and on line 540. I have never tried to figure it exactly (where would the fun be?), but it seems a winning strategy requires maximum horsepower and the stickiest tires; pit stops just don't seem to cost that much. I'd say the game needed new parameters if it weren't for the fact that NASCAR racing seems to operate just about that way.

Driving style does present a clear dilemma. In this game, preachers just don't win except through attrition. On the other hand, the driver who is always "belly to the wall" is going to be in contention while he lasts. With a mean distance to crash of 550 miles, he even has an apparent statistical chance of survival. Since the distribution is almost exponential, however, it is not awfully likely that he'll even reach the mean (the mean of the exponential is dragged upward by the few who go on forever; the mode is a hair above zero). Again, if you know a better way, the statements you're looking for are on lines 540 and 4010.

I tried to write this in as minimal a BASIC as possible. The only system supplied functions used are RND, ABS, LOG, SQR, INT, and COS. Our RND is a bit strange, so you may need a different initialization (e.g. RANDOM) than that found on line 150. The program also uses GOSUBS and statement functions.

The game was originally written for a WANG 3300 configuration that is notoriously intolerant of large programs (with about 4K available to store *and* execute the program, it's understandable). That version did not allow any change in strategy, which decreases the level of interest somewhat. A discussion with Dave Ahl at the CCUC in Fort Worth led to the present version, which is much more fun. I may even try to get this one onto the WANG (it currently is on a Univac Spectra 70/7).

```
100 REM*****GAME "DAYTONA 500"*****
110 REM*****BY GEOFFREY CHURCHILL*****
120 REM*****GEORGIA STATE UNIVERSITY*****
130 REM
140 REM  THIS IS HOW OUR RND IS INITIALIZED
150 X=RND(-1)
160 DIM E(14,2),S(6),G(6),Q(6),R(6,2),D(6),T(6),H(6)
170 DEF FNZ(X)=SQR(-2*LOG(RND(0)))*COS(6.283185*RND(0))*X
180 PRINT "WELCOME TO THE DAYTONA 500."
190 PRINT "DO YOU NEED INSTRUCTIONS (1=YES)";
200 INPUT Y1
210 IF Y1<>1 THEN 290
220 PRINT "HORSEPOWER AFFECTS SPEED AND TIRE MILEAGE; MAY BE 350 TO 700."
230 PRINT "TIRE COMPOUND AFFECTS THEM TOO. SELECT ";
240 PRINT "SOFT(1),MEDIUM(2),OR HARD(3)."
250 PRINT "SOFT COMPOUNDS GO FASTER BUT WEAR OUT SOONER."
260 PRINT "YOUR DRIVING STYLE AFFECTS SPEED AND RISK. YOU MAY CHOOSE"
270 PRINT "DRIVING STYLE PREACHER(1),BANKER(2),PROFESSOR(3),"
280 PRINT "OIL WILDCATTER(4),TAXI DRIVER(5),OR BTTW(6)."
290 PRINT "INPUT LIKE 400,3,6"
300 E(1,1)=14
310 E(1,2)=1E30
320 DEF FNC(C1)=(C1+2)^2+C1-3
330 K9=6
340 FOR C1=1 TO 6
350 PRINT "CAR #";FNC(C1);"HP, TIRES, STYLE";
360 INPUT H1,T1,S1
370 H(C1)=H1
380 REM  EXPECTED LAP SPEED
390 S(C1)=7*SQR(350+.5*H1)-T1+S1
400 REM  EXPECTED MILES/TANK
410 G(C1)=300-100*H1/350
420 REM  EXPECTED MILES/SET OF TIRES
430 R(C1,1)=200*T1-2*S1-H1/100
440 R(C1,2)=R(C1,1)+FNZ(5)
450 D(C1)=0
460 T(C1)=0
470 E2=G(C1)+FNZ(5)
480 IF E2<R(C1,2) THEN 500
490 E2=R(C1,2)
500 E1=6+C1
510 E2=INT(E2/2.5+.5)*2.5
520 GOSUB 2010
530 REM  MILEAGE AT CRASH IF NO STRATEGY CHANGE...IF>500, NO CRASH!
540 E2=INT((-1000+75*S1)*LOG(RND(0))+.5)
550 E1=C1
560 GOSUB 2010
570 NEXT C1
580 E1=13
590 E2=50
600 GOSUB 2010
610 PRINT "THE GREEN FLAG IS OUT, AND THERE THEY GO!"
620 GOSUB 3010
630 IF E1=13 THEN 1080
640 IF E1=14 THEN 1080
650 IF E1<7 THEN 1000
660 IF S(E1-6)<0 THEN 620
670 PRINT "CAR #";FNC(E1-6);"HAS PITTED AFTER";E2/2.5;"LAPS"
680 C1=E1-6
690 S1=S(C1)+FNZ(2)
700 T(C1)=T(C1)+(E2-D(C1))/S1
710 D(C1)=E2
720 PRINT "CHANGE STRATEGY (0=NO)";
730 INPUT Y1
740 IF Y1=0 THEN 820
750 PRINT "INPUT TIRES,STYLE";
760 INPUT T1,S1
770 H1=H(C1)
780 S(C1)=7*SQR(350+.5*H1)-T1+S1
790 G(C1)=300-100*H1/350
800 R(C1,1)=200*T1-2*S1-H1/100
810 GOSUB 4000
820 PRINT "PIT STOP IS FOR GAS ";
830 S1=15+FNZ(3)
840 E2=E2+G(C1)+FNZ(5)
850 IF Y1=1 THEN 870
860 IF D(C1)+100<R(C1,2) THEN 900
870 S1=S1+6+FNZ(2)
880 PRINT " AND TIRES"
890 R(C1,2)=D(C1)+R(C1,1)+FNZ(5)
900 PRINT
910 IF R(C1,2)>=E2 THEN 930
920 E2=R(C1,2)
930 S1=(S1+2*ABS(Y1))/3600
940 T(C1)=T(C1)+S1
950 PRINT "CAR #";FNC(C1);"WAS IN THE PITS";INT(S1*36000)/10;"SECONDS"
960 E1=C1+6
970 E2=INT(E2/2.5+.5)*2.5
980 GOSUB 2010
990 GOTO 620
1000 S(E1)=-100
1010 PRINT "CAR #";FNC(E1);"HAS CRASHED AT";E2;"MILES";
1020 PRINT " AND IS OUT OF THE RACE"
1030 K9=K9-1
1040 IF K9<1 THEN 1060
1050 GOTO 620
1060 PRINT "ALL THE EARLY LEADERS HAVE CRASHED.  CAR #";FNC(7);"WINS."
1070 GOTO 5000
1080 S3=1000
1090 FOR C1=1 TO 6
1100 IF S(C1)<=0 THEN 1160
1110 Q(C1)=S(C1)+FNZ(2)
```

```
1120 T1=T(C1)+(E2-D(C1))/Q(C1)
1130 IF T1>S3 THEN 1160
1140 S3=T1
1150 C2=C1
1160 NEXT C1
1170 PRINT
1180 PRINT
1190 PRINT "AT";E2;"MILES, CAR #";FNC(C2);"IS LEADING."
1200 FOR C1=1 TO 6
1210 IF S(C1)<=0 THEN 1290
1220 L1=E2-(D(C1)-Q(C1)*(T(C1)-S3))
1230 IF T(C1)>S3 THEN 1270
1240 D(C1)=D(C1)+Q(C1)*(S3-T(C1))
1250 T(C1)=S3
1260 L1=E2-D(C1)
1270 IF C1=C2 THEN 1290
1280 PRINT "CAR #";FNC(C1);"TRAILS BY";INT(10*L1)/10;"MILES"
1290 NEXT C1
1300 PRINT "LEADER'S AVERAGE SPEED IS";E2/T(C2);"MPH"
1310 E2=E2+50
1320 IF E2<=500 THEN 1360
1330 PRINT "THE DRIVER OF CAR #";FNC(C2);"IS THE NEW";
1340 PRINT " DAYTONA 500 CHAMPION AT";500/T(C2);"MPH!"
1350 GOTO 5000
1360 GOSUB 2010
1370 GOTO 620
2000 REM  GOSUB TO STORE NEW EVENTS
2010 FOR I=1 TO 14
2020 IF E2<=E(I,2) THEN 2060
2030 NEXT I
2040 PRINT "DAMN!"
2050 GOTO 5000
2060 FOR J=14 TO I+1 STEP -1
2070 E(J,1)=E(J-1,1)
2080 E(J,2)=E(J-1,2)
2090 NEXT J
2100 E(I,1)=E1
2110 E(I,2)=E2
2120 RETURN
3000 REM  GOSUB TO RETRIEVE NEXT EVENT
3010 E1=E(1,1)
3020 E2=E(1,2)
3030 FOR I=1 TO 13
3040 E(I,1)=E(I+1,1)
3050 E(I,2)=E(I+1,2)
3060 NEXT I
3070 RETURN
4000 REM  GOSUB TO CHANGE CRASH MILEAGE
4010 S1=(-1000+75*S1)*LOG(RND(0))+.5
4020 FOR I=1 TO 14
4030 IF E(I,1)=C1 THEN 4070
4040 NEXT I
4050 PRINT "DAMN!"
4060 GOTO 5000
4070 E1=C1
4080 E2=INT((D(C1)/500)*E(I,2)+((500-D(C1))/500)*S1)
4090 IF E2>D(C1)+1 THEN 4110
4100 E2=D(C1)+INT(10*(RND(0))+1)
4110 FOR J=I TO 13
4120 E(J,1)=E(J+1,1)
4130 E(J,2)=E(J+1,2)
4140 NEXT J
4150 GOSUB 2010
4160 RETURN
5000 END
```

```
  RUN
WELCOME TO THE DAYTONA 500.
DO YOU NEED INSTRUCTIONS (1=YES)?0
INPUT LIKE 400,3,6
CAR # 7  HP, TIRES, STYLE?700,3,6
CAR # 15 HP, TIRES, STYLE?700,2,6
CAR # 25 HP, TIRES, STYLE?700,1,6
CAR # 37 HP, TIRES, STYLE?700,3,6
CAR # 51 HP, TIRES, STYLE?700,2,6
CAR # 67 HP, TIRES, STYLE?700,1,6
THE GREEN FLAG IS OUT, AND THERE THEY GO!


 AT 50 MILES, CAR # 7 IS LEADING.
CAR # 15 TRAILS BY 1 MILES
CAR # 25 TRAILS BY 1.3  MILES
CAR # 37 TRAILS BY 1.5  MILES
CAR # 51 TRAILS BY .7   MILES
CAR # 67 TRAILS BY 1.4  MILES
LEADER'S AVERAGE SPEED IS 192.699   MPH
CAR # 51 HAS PITTED AFTER 37  LAPS
CHANGE STRATEGY (0=NO)?0
PIT STOP IS FOR GAS
CAR # 51 WAS IN THE PITS 15.7 SECONDS
CAR # 67 HAS PITTED AFTER 39  LAPS
CHANGE STRATEGY (0=NO)?0
PIT STOP IS FOR GAS  AND TIRES

CAR # 67 WAS IN THE PITS 19.9 SECONDS
CAR # 7  HAS PITTED AFTER 39  LAPS
CHANGE STRATEGY (0=NO)?0
PIT STOP IS FOR GAS
CAR # 7  WAS IN THE PITS 11.8 SECONDS


 AT 100   MILES, CAR # 15   IS LEADING.
CAR # 7  TRAILS BY .5   MILES
CAR # 25 TRAILS BY 1.2  MILES
CAR # 37 TRAILS BY 1.5  MILES
CAR # 51 TRAILS BY .7   MILES
CAR # 67 TRAILS BY 1.9  MILES
LEADER'S AVERAGE SPEED IS 189.738   MPH
CAR # 37 HAS PITTED AFTER 40  LAPS
CHANGE STRATEGY (0=NO)?0
PIT STOP IS FOR GAS
CAR # 37 WAS IN THE PITS 12.6 SECONDS
CAR # 15 HAS PITTED AFTER 41  LAPS
CHANGE STRATEGY (0=NO)?0
PIT STOP IS FOR GAS
CAR # 15 WAS IN THE PITS 11.4 SECONDS
CAR # 25 HAS PITTED AFTER 42  LAPS
CHANGE STRATEGY (0=NO)?0
PIT STOP IS FOR GAS  AND TIRES

CAR # 25 WAS IN THE PITS 15.4 SECONDS


AT 150   MILES, CAR # 15   IS LEADING.
CAR # 7  TRAILS BY .5   MILES
CAR # 25 TRAILS BY 2 MILES
CAR # 37 TRAILS BY 3 MILES
CAR # 51 TRAILS BY .2   MILES
CAR # 67 TRAILS BY 1.6  MILES
LEADER'S AVERAGE SPEED IS 189.301   MPH
CAR # 67 HAS CRASHED AT 164   MILES AND IS OUT OF THE RACE
CAR # 7  HAS CRASHED AT 180   MILES AND IS OUT OF THE RACE


AT 200   MILES, CAR # 15   IS LEADING.
CAR # 25 TRAILS BY .7   MILES
CAR # 37 TRAILS BY 2.3  MILES
CAR # 51 TRAILS BY .1   MILES
LEADER'S AVERAGE SPEED IS 188.992   MPH
CAR # 37 HAS PITTED AFTER 80  LAPS
CHANGE STRATEGY (0=NO)?0
PIT STOP IS FOR GAS
CAR # 37 WAS IN THE PITS 13.7 SECONDS
CAR # 15 HAS PITTED AFTER 81  LAPS
CHANGE STRATEGY (0=NO)?0
PIT STOP IS FOR GAS
CAR # 15 WAS IN THE PITS 20.4 SECONDS
CAR # 51 HAS PITTED AFTER 83  LAPS
CHANGE STRATEGY (0=NO)?0
PIT STOP IS FOR GAS
CAR # 51 WAS IN THE PITS 14.3 SECONDS
CAR # 25 HAS PITTED AFTER 84  LAPS
CHANGE STRATEGY (0=NO)?0
PIT STOP IS FOR GAS  AND TIRES

CAR # 25 WAS IN THE PITS 14.4 SECONDS


 AT 250   MILES, CAR # 15   IS LEADING.
CAR # 25 TRAILS BY .9   MILES
CAR # 37 TRAILS BY 3.2  MILES
CAR # 51 TRAILS BY .5   MILES
LEADER'S AVERAGE SPEED IS 188.447   MPH
CAR # 37 HAS PITTED AFTER 119 LAPS
CHANGE STRATEGY (0=NO)?1
INPUT TIRES,STYLE?1,6
PIT STOP IS FOR GAS  AND TIRES

CAR # 37 WAS IN THE PITS 20   SECONDS


 AT 300   MILES, CAR # 15   IS LEADING.
CAR # 25 TRAILS BY 1.2  MILES
CAR # 37 TRAILS BY 4.1  MILES
CAR # 51 TRAILS BY .7   MILES
LEADER'S AVERAGE SPEED IS 188.543   MPH
CAR # 25 HAS PITTED AFTER 122 LAPS
CHANGE STRATEGY (0=NO)?1
INPUT TIRES,STYLE?3,6
PIT STOP IS FOR GAS  AND TIRES

CAR # 25 WAS IN THE PITS 29   SECONDS
CAR # 51 HAS PITTED AFTER 123 LAPS
CHANGE STRATEGY (0=NO)?1
INPUT TIRES,STYLE?1,6
PIT STOP IS FOR GAS  AND TIRES

CAR # 51 WAS IN THE PITS 19.7 SECONDS
CAR # 15 HAS PITTED AFTER 124 LAPS
CHANGE STRATEGY (0=NO)?1
INPUT TIRES,STYLE?1,5
PIT STOP IS FOR GAS  AND TIRES

CAR # 15 WAS IN THE PITS 24.8 SECONDS


 AT 350   MILES, CAR # 15   IS LEADING.
CAR # 25 TRAILS BY 2.6  MILES
CAR # 37 TRAILS BY 3.5  MILES
CAR # 51 TRAILS BY .6   MILES
LEADER'S AVERAGE SPEED IS 188.32 MPH


AT 400   MILES, CAR # 15   IS LEADING.
CAR # 25 TRAILS BY 4 MILES
CAR # 37 TRAILS BY 5.2  MILES
CAR # 51 TRAILS BY 1.1  MILES
LEADER'S AVERAGE SPEED IS 188.808   MPH


 AT 450   MILES, CAR # 15   IS LEADING.
CAR # 25 TRAILS BY 2.9  MILES
CAR # 37 TRAILS BY 4.2  MILES
CAR # 51 TRAILS BY 0 MILES
LEADER'S AVERAGE SPEED IS 188.719   MPH
CAR # 51 HAS CRASHED AT 461   MILES AND IS OUT OF THE RACE
CAR # 15 HAS PITTED AFTER 190 LAPS
CHANGE STRATEGY (0=NO)?0
PIT STOP IS FOR GAS  AND TIRES

CAR # 15 WAS IN THE PITS 15.7 SECONDS
CAR # 37 HAS PITTED AFTER 190 LAPS
CHANGE STRATEGY (0=NO)?0
PIT STOP IS FOR GAS  AND TIRES

CAR # 37 WAS IN THE PITS 17.4 SECONDS


 AT 500   MILES, CAR # 15   IS LEADING.
CAR # 25 TRAILS BY 1.8  MILES
CAR # 37 TRAILS BY 4.7  MILES
LEADER'S AVERAGE SPEED IS 188.525   MPH
THE DRIVER OF CAR # 15  IS THE NEW DAYTONA 500 CHAMPION AT 188.525
MPH!
```

Reviews

Sophisticated Electronic Pocket Calculators: Theory and Practice for the Consumer and User

Edward R. Tufte*

Here are nine principles to help the consumer and user of sophisticated pocket calculators. These principles are general, applying to most all calculators (at least those that have a reasonable number of keyboard operations—say 40 or more).

I have no personal interest, direct or indirect, in any of the calculators or their parent companies discussed here. Among the calculators I have purchased, three were pretty good and two were lemons. Among those borrowed, the good-to-lemon ration has also run about 3 to 2. At any rate, this review is an independent personal evaluation, reflecting solely my experiences and prejudices.

GENERAL PRINCIPLE 1: *The half-life of calculator prices is about 18 months.* In other words, today's price will be cut by at least 50% within the next 18 months. The Texas Instruments SR-51, for example, was advertised at $224.95 in the March, 1975 issue of *Scientific American.* In February, 1977 the same machine sells for $65 at discount shops. General Principle 1 has several consequences:

Don't buy a calculator when it first comes on the market, unless you really like it. Consider if you had bought the SR-51 at $225 two years ago. Since it is now selling at $65, you would have paid $160 to rent that machine for two years plus have to forego other uses of the $225 for the whole period.

Buy at a discount. There is a lively, fast-moving market in calculators and the short half-life of prices encourages big discounts. Discounts of a minimum of 20% from the list price are available. The nominal price given by the manufacturer in the glossy advertisements is pretty much fantasy. By the way, the guarantee on the calculator is with the *manufacturer* and so there are no advantages to local servicing because there is no local servicing (Sam's Camera and Calculator Shop can't fix an HP-65 programmable, magnetic-strip read/write calculator).

GENERAL PRINCIPLE 2: *The breakdown rate of pocket electronic calculators is too high.* What evidence I have indicates that substantial numbers of calculators die within the first year of operation—perhaps one-third of all machines. There is too much of a throw-away mentality prevailing in the industry. Hewlett-Packard has the best reputation for reliable calculators (which may account for their relatively high cost).

GENERAL PRINCIPLE 3: *Calculators will continue to improve at the rate of the past few years.* A new generation has passed about every six years in the development of computational devices. On average, each new generation has increased speed by tenfold, memory capacity twentyfold, decreased component cost tenfold, and system cost at least two fold. Rapid progress continues. Now 16,000 bits of binary storage are available on a 1/32-inch square—just what you've always wanted. The rate of improvement means that today's machine will be replaced by something twice as good at half the price next year. It also means that calculators should be rapidly depreciated on your income tax.

GENERAL PRINCIPLE 4: *Computational technology has completely overrun input-output technology.* The great limits on calculators for any sort of serious work are the inability to monitor past inputs, and the single read-out register. We have a 19th-century printing technology that cannot cope with a 21st-century computational technology.

---

*Edward R. Tufte is Professor of Public Affairs at Princeton University. His books include *Data Analysis for Politics and Policy* and *Elections and Economics.*

GENERAL PRINCIPLE 5: *Printing is worth it.* The great tragedy of the HP-65, and $800 programmable wonder-machine, is that it shows you only one number at a time, often only one time. For what most of us do with calculators, we want to see a lot of numbers a lot of times. Printing, even though expensive, is worth it. I would buy a much less computationally fancy machine in order to have one of those little Mickey Mouse printers now available.

The philosophy behind most calculators today is one that was commonly found in computer centers some years back: the point of the machine is to do lots of fancy computing in order to come up with an answer consisting of a single number. Such doctrine, however is not consistent with the development of good home calculating.

GENERAL PRINCIPLE 6: *Sophisticated pocket calculators, particularly the programmable kind, are like those phonograph records that purport to teach foreign languages (Learn Swedish in 8 Hours); that is, they are purchased with all kinds of good intentions to really make use of them and change one's life, they are used once or twice, and then they sit on the shelf months on end just making one feel guilty.* Programmable calculators with 224 steps and read/write options are nifty but expensive; make sure there is at least one chance in ten that you will use the programmable part of the package after you buy such a machine. Machines that print, rather than those that are programmable, are much more likely to be useful. (Programmable calculators can even cram into their little memories a multiple regression program for three variables. Terribly ingenious, but not useful for any serious analysis, it is like Dr. Samuel Johnson's dog that could walk on its hind legs: "It is not done well, but you are surprised to find it done at all.")

GENERAL PRINCIPLE 7: *Instruction manuals vary tremendously in quality; and they usually have errors in them.* Sometimes instruction manuals for calculators appear to have been written originally in some language other than English—and both the author and translator had something more important to do that day than produce the manual. Hewlett-Packard manuals are easily the best; those from Texas Instruments are pretty good, but uneven; and manuals from other companies are a real risk. I recommend looking at the instruction manual before buying any brand except HP and TI.

GENERAL PRINCIPLE 8: *Calculators are designed by engineers and business people for engineers and business people.* Calculator manufacturers believe that their market is found among people in business and engineering. Machines are not designed to handle problems of data analysis and simple statistical work; data files are hard to manage; statistical manipulations are hard to perform.

GENERAL PRINCIPLE 9: *By any sort of long-term perspective, the small sophisticated electronic pocket calculator is a miracle.* No telling what is good for, but it is still a miracle. For a few hundred dollars, I have as much computational power on my desk now as there was in most major university computing centers 15 or 20 years ago. If I only knew what to do with it. ■

DATA PROCESSING DEFINITIONS

ASSUMED DECIMAL POINT. Located two positions to the right of a programmer's current salary in estimating his own worth.

BIT. The increment by which programmers slowly go mad.

CHAINING. A method of attaching programmers to desks to speed up output.

CHECKPOINT. The location from which a programmer draws his salary.

COMMON LANGUAGE. The first thing a programmer must forget in order to be successful.

CORE STORAGE. A receptacle for the center section of apples.

COUNTER. A device over which martinis are served.

ERROR. What someone else has made when he disagrees with your computer output.

EXTERNAL STORAGE. Wastebasket.

FIXED WORD LENGTH. Four-letter words used by programmers in a state of confusion.

FLOATING CONTROL. A characteristic exhibited when you have to go to the restroom but can't leave the computer.

FLOATING POINT. The absolute limit before floating control is lost.

FLOW CHART. A graphic representation of the fastest route to the restroom.

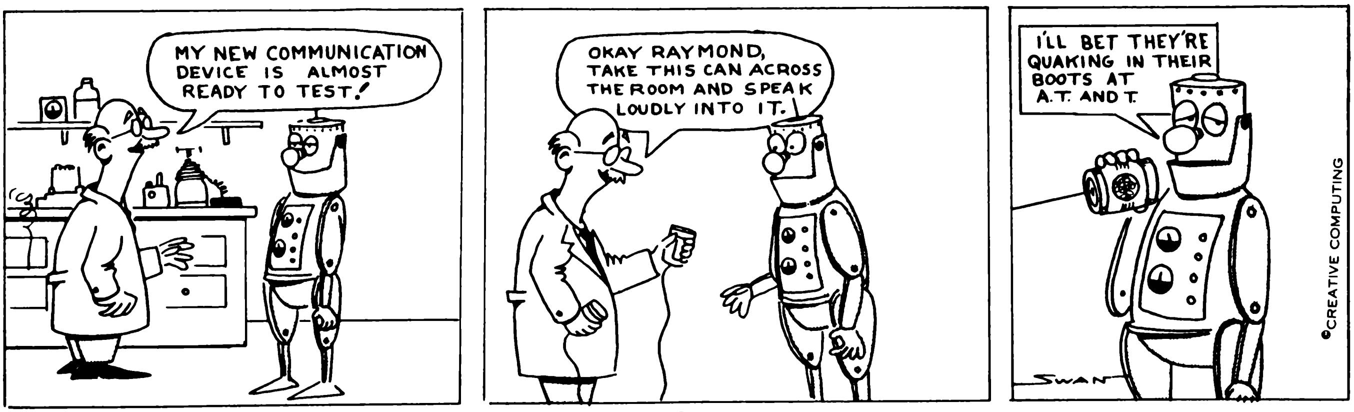
INPUT. Food, whiskey, beer, aspirin, etc.

MACRO. The last half of an expression of surprise: "Holy Macro."

MEMORY DUMP. Amnesia.

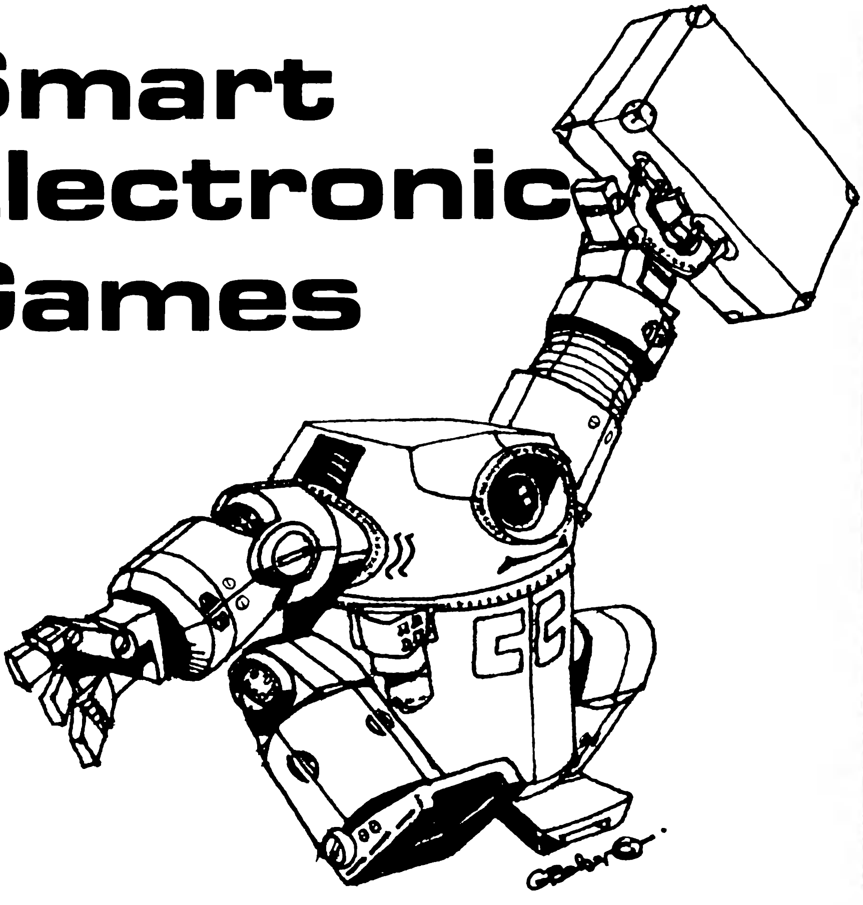
PROGRAMMER. Red-eyed, mumbling mammal capable of conversing with inanimate objects.

—Modern Data


MY NEW COMMUNICATION DEVICE IS ALMOST READY TO TEST!
OKAY RAYMOND, TAKE THIS CAN ACROSS THE ROOM AND SPEAK LOUDLY INTO IT.
I'LL BET THEY'RE QUAKING IN THEIR BOOTS AT A.T. AND T.
SWAN

©CREATIVE COMPUTING

Smart Electronic Games



As anyone who has visited a department store or read a mail-order catalog recently knows, recreation and leisure-time options are rapidly expanding. One little corner of this market, perhaps the fastest-growing single segment, incorporates solid-state electronics into video games, stand-alone games, and calculator variants. In the past *Creative* has taken some brief looks at video games ("Playing Pong to Win," May/Jun 75; "Odyssey Video Games," Nov/Dec 76). Over the next several issues we'll be taking an in-depth look into the mysteries of a number of the new stand-alone games (see the article on Comp IV elsewhere in this issue).

For those of you interested in what's commercially available for gift-giving this holiday season, here is a roundup (undoubtedly incomplete) of some of the new smart electronic games. The games are widely discounted, but not nearly to the extent of hand-held calculators. Also, we found some stores selling them at higher-than-list prices! A word of warning: quality control may not be up to snuff — one of the games we obtained quit in the first 5 minutes. But most worked OK and were generally lots of fun! — DHA

Mattel Auto Race

Sliding control steers car (bright light blip) to avoid hitting computer-controlled oncoming cars (dimmer light blips). At end of lap, car will begin new lap. Gear-shift controls speed. Simulated engine sounds become louder as speed increases. Elapsed time shown. On collision, beep sounds and your car is pushed back a space or two. Complete the 4 laps in 99 seconds or less, you win and a pulsating beep is emitted. If you fail to do so, clock stops at 99; constant beep is heard.

Beige plastic case 1x3x5 in. Uses one 9-volt battery. Retail range, $20 to $25.

MATTEL ELECTRONICS
Auto Race
GEAR
39

MATTEL ELECTRONICS
PLAY STATUS
PUNT
SCORE
FOOTBALL

Mattel Football

You control running back (bright light blip) through the computer-controlled defensive tacklers (dimmer light blips). If running back hits a tackler, a simulated whistle sounds. ST (status) key shows new down, yard line and no. of yards for first down. SC (score) key reveals both teams' scores and time remaining. On scoring, you hear the "victory charge." K (kick) key is used for punting and field goals. Game time is 10 min. Pro 1/Pro 2 switch allows you to choose average or expert opposition.

Beige plastic case, 1x3x5 in. Uses one 9-volt battery. Retail range, $30 to $35.

Mattel Missile Attack

It's you against the computer as you defend your city! The computer launches missiles aimed to destroy your city, and you launch anti-missile missiles to intercept and destroy the incoming barrage. The more you destroy, and the higher in the sky you do so, the more points you get. If you miscalculate and an attacking missile makes a direct hit on your city, the computer plays a mini rendition of "Taps"!

Blue-grey plastic case, 1x3x5 in. Uses one 9-volt battery. Retail range, $20 to $25.

MATTEL ELECTRONICS
MISSILE ATTACK
FIRE
ARM
OFF
RADAR

CHESS
CHALLENGER

Fidelity Electronics Chess Challenger

The board is defined by numbers along the X-axis and letters on the Y-axis. Moves are entered on the calculator-like keyboard by defining the current location of a piece and where you want to move it to. You may castle or capture en passant, although the computer isn't programmed to use the latter move. Within seconds of your move, the computer analyzes possible counter-moves and responds with its best choice. It can be beaten by an average player from 25% to 75% of the time.

Experts who want to play a more difficult game can send their unit, along with $75, to the manufacturer for upgrading with a more complex ROM, which increases the difficulty level, has three levels of play. A newer model, which has three levels of play and is essentially the same as an upgraded basic model, but with some cosmetic changes, is $275.

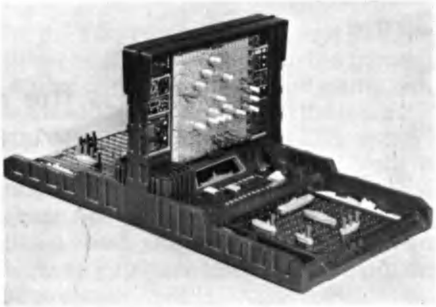
Walnut wood housing, 12x8x1 in. Operates on line current. Retail range, $160 to $200.

NUMBER SEQUENCE
COMP IV

Milton Bradley Comp IV

Using the calculator pad, you try to guess a secret 3-digit number (like Bagels), 4-digit one (Mastermind) or 5-digit one. Comp IV tells you how many digits are correct and how many are in the correct position. Clever deduction will allow you to get the number in the fewest possible guesses. Comp IV prods you with flashing lights if you delay too long. Use a different number each round, or the same one if two or more players are competing against each other.

Plastic console, 7½x4x4 in. Uses one 9-volt battery. Retail range, $20 to $40.



Milton Bradley Electronic Battleship

Two players, acting as fleet commanders, determine the best locations for each of their ships in this game of electronic hide-and-seek on the high seas. The continuous sound of sonar adds realism to the mission, as you set your console controls to fire on sectors of the board where the opponent's ships may be hidden. Press the Fire button, and you hear the whoosh of a torpedo heading for its target; hits are indicated by a flash of light on the scope and the sound of a realistic explosion. When you've sunk your opponent's entire fleet, Electronic Battleship sounds a continuous whoop of victory.

Plastic board and display, 21x10¾x9 in. Uses four 9-volt batteries. Retail range, $30 to $50.

Parker Brothers Code Name: Sector

The human players (1 to 4) are destroyer commanders, moving their ships among 4800 grids on a nautical chart. The microprocessor stores, computes, and displays speed, direction, and location of each ship. It also moves the sub, keeping its location secret but giving information that can be used to track and attack it. But be careful — that sub can torpedo destroyers! There are eleven control buttons. Comes complete with 8 replica subs for scoring, facsimile Navy parallel rule, tracking crayons, wiper, and sharpener. Not a simple game! Age 12 and up.

Plastic board and display, 19x13½x4 in. Uses one 9-volt battery. Retail range, $30 to $45.

A243
43F
763

Unisonic Vegas 21

Casino blackjack with full Las Vegas rules. Your initial stake can be up to $50 million; before each hand you designate your wager. Your winnings and losses are posted after each hand. You can make insurance bets, split pairs, double down. Automatic deck shuffle after 41st card is dealt. Doubles as a standard calculator.

Desk-top version woodgrain and gray plastic, 7x9x4 in. Large L.E.D. readout. Uses 3 C-cells. Retail range, $39 to $59.

Hand-held version chrome and black plastic, 1x3x5 in. Uses 3 AA batteries. Retail range, $30 to $50.

FOOTBALL
LUNAR LANDER
MATHEMAGICIAN
LET HIM TEACH YOU
1234567890

APF Mathemagician

A teaching calculator and device for game-playing all in one. Simple or complex 1- or 2-digit problems can be performed. Four-function calculator does multiplication, division, addition and subtraction. In addition, six games can be played — Gooey Gumdrop, The Number Machine, Walk-the-Plank, Football, Lunar Lander and Countin' On, each with its own plastic overlay. All games can be played by one or two people. In "Lunar Lander," you take over the controls of a lunar landing module at an altitude of 300 ft. above the moon's furface. Try and land the module safely! Switch to "Gooey Gumdrop" and try to find the secret location of the gumdrop bomb before it explodes. In "Walk the Plank," you try to guess the machine's secret number in three tries. If you lose, you walk the plank!

Black and gray plastic case, 8½x5x1½ in. Uses one 9-volt battery. $39.95. ■

Profile of a Smart Electronic Game . . .

Guess the random number generated by this new game, which gives hints.

COMP IV

Stephen B. Gray

"I am programmed to beat you," says the ad, and continues, in smaller print, "You are mortal. I am the product of millions of dollars of research. You will attempt to deduce the numbers I have hidden in my computer memory. I will reveal only which of your digits are correct and how many are in the right order. As your excitement mounts, you will get (ugh!) emotional. I will not. You are mortal. I am Comp IV, the new electronic challenge game with 32,000 number combinations, from Milton Bradley."

Description

Despite the provocative challenge, Comp IV looks quite simple and harmless. Weighing only about eleven ounces and fitting comfortably into one hand, it consists outwardly of no more than a blue plastic case with a 12-button keyboard, two vertical rows of five LEDs (light-emitting diodes) on a sloping display panel, an on/off switch, and a compartment for a nine-volt battery.

Inside, Comp IV looks almost as simple. Mounted under the 12 keys is a very thin Texas Instruments X-Y matrix on a plastic substrate. Pressing a key causes an X (row) wire to touch a Y (column) wire. The matrix is separated from the 12 keys by a thin sheet of plastic foam and a sheet of Mylar, mostly to protect the keyswitch matrix from anything that small (or large) hands might spill onto the keyboard.

The only other item concealed inside the case is the brains of the game, a 28-pin IC, also from TI, a one-chip microcomputer in the TMS1000 series. This family of ICs is used in every one of the TI calculators, as well as in another Milton Bradley game, Electronic Battleship, which will be reviewed at a later date

The TMS1000 series of microcomputers sells for less than $3 in quantity, and is recommended by TI for use in coin changers, fuel metering, oven control, games, automatic telephone answering, and many other applications. Each member of the family contains registers, a program counter, an arithmetic logic unit, RAM and ROM memory, I/O circuitry, and a clock. The particular one used in Comp IV contains 1,024 8-bit words of ROM memory for the random-number generator and the game-play algorithm, plus 64 4-bit RAM words to hold the numbers entered from the keyboard.

Basic Operation

After turning on Comp IV, you can check out the circuits by pressing the keys in the sequence 7-E-1-2-3-4-E (the 12 keys are labelled 0 through 9, plus R for Reset and E for Enter). If the unit is working correctly, the five lights in the left vertical row (Number) will flash along with the four bottom lights in the right vertical row (Sequence). The top right LED is the Ready indicator.

Press R (with or without having previously gone through the checkout procedure), and the LEDs will flash in seven preset combinations (the manual says Comp IV is scanning while it selects a number, but no doubt the number is selected long before the first combination of lights goes on. The flashing-light combinations simply make it look more like a computer). Then the Ready light goes on, to indicate that Comp IV is ready to accept your guess as to what number it has generated. In the numbers generated by Comp IV, no digit is repeated, so that the secret number will never be 553, for instance. That would make the game just a little too difficult for most people.

Your first entry determines the complexity of that particular game. If you wish to play a three-digit game, make your first guess a three-digit one, and enter it into the computer by pressing the E key. If you wish to play a four-digit or a five-digit game, make a first guess of four or five digits, before pressing E.

When you enter your guess, Comp IV compares it with the random number it has generated, and responds with lights on the sloping screen. The left row, marked NUMBER, indicates how many digits you guessed correctly; the right row, labelled SEQUENCE, shows how many of the digits guessed correctly are in the correct order (but not *which* numbers are in the right order).

As further entries are made, the display will help you figure out which digits to eliminate and which to confirm. If there is no match between your entry and Comp IV's number, the only light on the display will be the Ready light. If you enter too many or too few digits, the 1 NUMBER light will blink.

When you finally enter the exact number that Comp IV is holding, all the lights on the screen will flash (except the Ready light). To play another game, press R, wait for the seven sets of flashes, enter the number of digits you wish to play, and thus you start a new game against a new number generated by Comp IV.

A pad is provided for keeping track of your entries, as well as of digits you're sure are (or are not) in the hidden number.

Demonstration

Suppose Comp IV has generated 436. If your guess is 423, you'll get a 2 light in the NUMBER column because you got the 4 and 3 right, and a 1 light in the SEQUENCE column because you guessed the 4 in the right place. If your guess had been 432, you'd have a 2 light in the SEQUENCE column.

Extra Features

Comp IV has two features designed to keep you from dawdling. The R light "also serves as a timer to help you keep track of your game." After about 30 seconds, it will flash slowly, and after another 30 seconds, will begin to flash faster, and stay that way. Also, responses are displayed for only about 30 seconds, after which they go off and the ready light comes on.

But these features are only to remind you that time is passing; if you're playing alone, you can take all day to figure out Comp IV's hidden number. This is entirely unlike some of the other games that can be played alone (and which we'll be reviewing in future issues), where you have automobiles or an opposing team or missiles coming at you, and you've *got* to make fast decisions *right now*.

Fast decisions are usually required, however, in group play with Comp IV, unless you don't mind being called a slowpoke or an idiot or worse.

Group Play

According to the Comp IV manual, the game can be played by a group in two ways. In Rotation Play, each player makes an entry, announces it to the group, and uses a sheet of the entry pad to keep track of the game's progress. "For large groups of experienced players, the five-digit game is often best." The first person to guess the number that Comp IV holds is the winner.

In Repeat Play, advantage is taken of a circuit that allows Comp IV to hold a number for repeated games, so that each player gets a chance for a complete game on that number. Each player guesses the correct number in secret, then presses E instead of R. This will keep the same number in Comp IV, instead of generating a new one. The player solving the secret number in the least number of steps is the winner. (A counter to indicate how many times the E key is pressed would be a big help, by showing exactly how many steps were required to find the secret number, but this would also increase the price of the game.)

Observations

Obviously, Comp IV has to generate a five-digit number each time, because it doesn't know if you're going to play a three-, four-, or five-digit game. So it may come up with 45283, but if you're playing a three-digit game, you'll win with 452.

The three-digit game is quite simple. It can be played without looking at the sequence lights, although such a game takes a little longer without the hints provided by these lights. After a three-digit game, some pessimists may think Comp IV is a child's toy. Let them try a four-digit game, and especially a five-digit game. They may well require a game pad, which can shorten the game for those who find they just can't keep all the "sure" and "out" numbers in their heads. Even with a game pad, there will no doubt be people who simply give up in frustration.

There is a way to cheat. The manufacturer probably provided this "easy way out" to keep some frustrated player from taking a sledge hammer to his (or somebody else's) Comp IV, or even to keep him from banging his head against the wall. You can build up quite an anger against an 11-ounce box of plastic and metal, if it refuses to divulge its secret number despite all your banging on the keys, and will only blink its little lights.

The manufacturer doesn't call it cheating, but instead politely refers to it as "one strategy that can be used to discover whether certain digits are held." You enter a single digit as many times as there are digits in the game. If you're playing a three-digit game, and want to find out for sure if there's a 7 in the secret number, enter 777, and if 7 was included, you'll get a 3 light in number, and a 1 light in sequence. Press all the other numbers three times, and you'll find the other two. Now all you have to do is figure out the sequence.

The game is so fascinating that you'll soon decide to get an AC adapter, rather than use up a lot of 9-volt batteries.

The random-number generator provides a completely new number each time; there is no stored set of numbers in the IC. The number of possible five-digit combinations of ten digits, with no repeated digits, is 10x9x8x7x6, or 30,240. The possible combination of four digits is 10x9x8x7, or 5,040.

Comparison with Computer Games

In the BAGELS game *(101 BASIC Computer Games)* essentially the same information is provided by the host computer, which indicates with words instead of lights the same information (PICO: one digit is correct, but in the wrong place; FERMI: one digit is in the correct place; BAGLES: no digit is correct).

BULCOW *(101 BASIC Computer Games)* is a somewhat advanced version of BAGELS, known as Bulls and Cows (more popular in England than in the U.S.). "A BULL is scored for each correct digit in the correct position and COW for each correct digit but out of position."

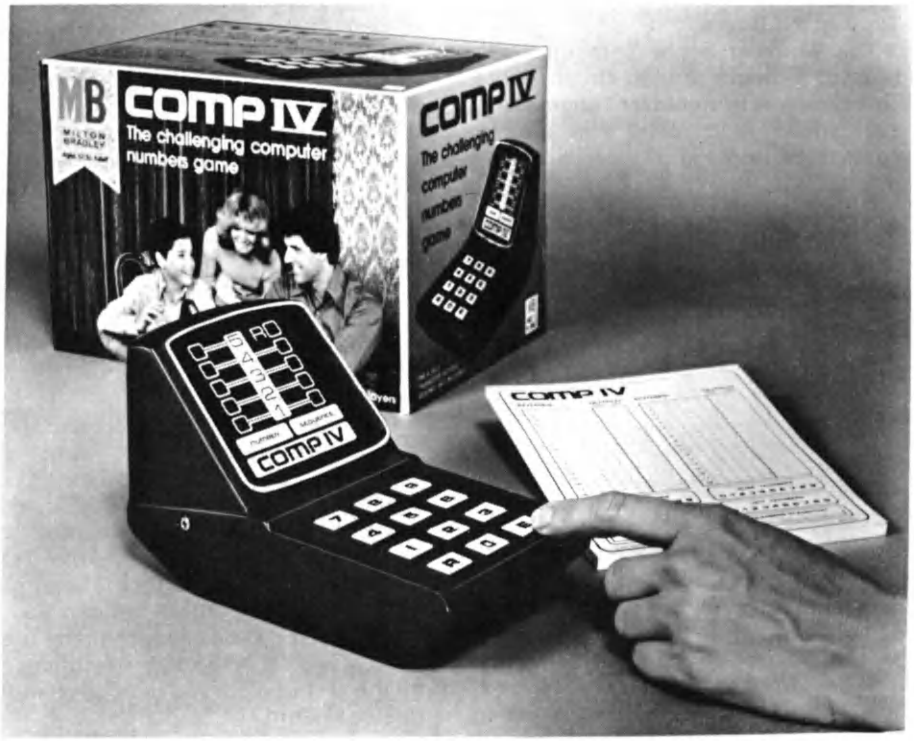
MASTERMIND (Mar-Apr 1976 *Creative Computing*) was originally "simply a commercial adaptation (using colors rather than numbers) of the game Bulls and Cows." Two people play against each other, using a plastic game board and plastic pegs of various colors. Each player tries to guess the color and position of his opponent's pegs concealed in a "hidden code" portion of the game board. *Creative* provides a computerized version.

MASTERBAGELS (Jan-Feb 1977 *Creative Computing*) combines Bagels, Bulls and Cows, Mastermind, and several other games "into a general deductive logic game." If the player wishes to play Bagels, he sets the initial parameters to N,3,9; if Mastermind is his choice, the setting is N,4,6. "But the real fun is trying new combinations."

History

The first Milton Bradley game came out in 1860, and is still available: "The Game of Life," involving growing up, college, a career, etc. (No relation to the game of Life invented in 1970 by Cambridge mathematician John Conway, and popularized by Martin Gardner in *Scientific American*. The whole game is ruled by three very simple genetic laws that define the survival, death and birth of the cells at each generation).

The 1960's opened the big era of plastic games, featuring hundreds of injection-molded items. In 1977, the Milton Bradley Company introduced their first electronic game, Comp IV. The price, depending on when and where you buy the game, is roughly between $20 and $40. ■


MB
MILTON BRADLEY
COMP IV
The challenging computer numbers game
COMP IV
The challenging computer numbers game
COMP IV
COMP IV

Selecting a Micro

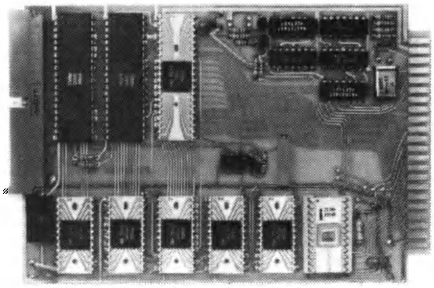
Stephen B. Gray

Selecting a hobby computer is getting harder every month, with new micros coming on the market all the time. Over 50 are available now, in kit or assembled form, or both, with a wide range of prices, and a great variety of features and peripherals. Some of the later ones have features that the earlier micros didn't, which may well cause some hobbyists to say, "I'll wait until a better machine comes along." That's a little like waiting for the *perfect* car, or stereo (or mate, for that matter). In waiting for that peak of perfection, you could lose out on a lot of fun and games (especially the games, in the case of micros).

The choice can be narrowed down by considering the various types of hobby computers. Five types dominate the market today, each with its own characteristics and appeal.

Type A: PC Board

The simplest hobby computer consists of a single printed-circuit board. This type was first sold for "engineering evaluation" (and many still are), to companies thinking of incorporating a microcomputer in one or more products. What you get is the central processor unit (CPU) board, which includes an MPU (microprocessor unit) as the main integrated circuit, along with control circuits, input/output interfacing, and perhaps some memory.


Type A: Wintek Control Module

Since the Type A micro is a minimum machine, you have to provide a power supply, facilities for input/output (such as a Teletype, or at least a keyboard and readout), and enough memory to allow the computer to run the programs you want, in the language of your choice.

With only IK or less of memory, your micro is limited to being programmed in machine language, or to short programs in assembly language. Some computerniks find both languages fascinating, but most hobbyists find them tedious and easy to make mistakes with. There's a lot to be learned in working up the ability to use an assembler fluently, but most of us are more interested in programming than in working that close to the hardware.

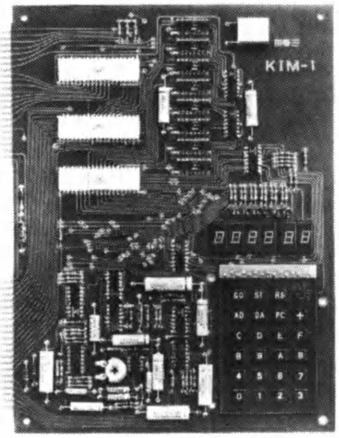
There's Tiny BASIC, a high-level language that makes programming much easier than with assembler, but if you try to use it with the limited amount of memory that comes with nearly all the Type A microcomputers, you won't have much memory, if any, to use for your own programs. This is because the Tiny BASIC itself takes up a certain amount of memory space, such as 2K. So you buy more memory, perhaps enough to run full-scale BASIC, for which you'd need at least 4K of memory for the BASIC interpreter itself, and at least 4K more for your own programs.

Examples of the Type A single-board computer are the Microcomputer Associates JOLT, and the National Semiconductor SC/MP.

Buy one of these Type A computers if you're more interested in hardware than software, are more interested in assembly language than in BASIC, and want to learn the fundamentals of computers. Type A is for the real computernik, as well as for the beginner who wants to learn all he can about hardware. It can be an inexpensive way to go, if you can keep down the prices of all the rest of what you need to create a usable system.

Type B: All on One Board

This type is one step up from the PC-board computer, because it adds an integral keyboard and display. The keyboard is almost always a hexadecimal type, with at least 16 keys labelled 0 to F, and perhaps several other keys for entering the keyboard data into memory, or displaying what's in a particular memory location, etc. The display in some of the cheaper Type B computers is made up of LEDs, which is not so easy to read; the majority provide a segmented alphanumeric display, with two, four, six or even more digits displayed.


Type B: MOS Technology KIM-1

Here is the most computer that can be bought for the least. Everything is supplied but the power supply, in nearly all cases, although some manufacturers provide one as an option.

With a hex keyboard and an alphanumeric display, the user can program in assembly language. Not very big programs, since most of the all-on-one-board computers are limited as to memory. But if you want more memory for your KIM-1 computer, you can get KIM-2, which is 4K, or KIM-3, which is 8K of memory. KIM-4 is a backplane into which memory (or other) boards can be plugged. For the EBKA 6502 Familiarizor, there's an expander board that will hold any or all of seven options, including a PROM programmer, 2K PROM memory, 4K RAM memory, baud-rate clock, and interfaces for dual cassette, serial and parallel operation.

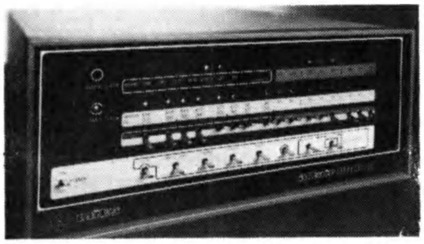
More of the Type B computers are available than any other type, and include the MOS Technology KIM-1, EBKA 6502 Familiarizor, E&L Mini-Micro-Designer, Intersil Intercept Jr., EPA-68, and the Martin Research Mike 3 and Mike 8. All but the last two consist of a single board; Mike 3 and Mike 8 are stacks of several boards, separated by spacers. This modular approach, with the keyboard and display on the top board, CPU on the second board, memory on the third, etc., permits using different CPU boards with differing MPU chips, such as the 8080A, Z-80, or 8008, for the hardware (and software) experimenter.

An all-on-one-board computer is recommended for the person who wants the most for the least, or the person who wants to learn the basics but who doesn't want to spend too much on something he may not use much, once he's learned enough to satisfy himself.

The same situation with languages applies to both Types A and B microcomputers. With limited memory you can work only in machine language or assembler. With a little more you can use Tiny BASIC, and with 4K or more, you can start working in standard BASIC.

Type C: Box With Lights and Switches

The first hobby computer to achieve stardom was the MITS Altair 8800, which had a full front panel, with two dozen switches and three dozen lights. Most of the switches are for addressing memory and for inserting or retrieving data from memory; the rest are function switches. Most of the lights indicate the address or the data; the rest indicate functions.



Type C: MITS Altair 8800b

The Altair 8800b, the latest model of the 8800, includes some additional function switches. Several other hobby computers are compatible with the Altair 8800b; that is, a circuit card from any one of them will plug into the motherboard of any other. This includes the Imsai 8080, PolyMorphic Poly 88, and the Processor Technology SOL, which use what has come to be called either the "Altair bus" or the "S-100 bus" (we call it the Altair S-100 bus, just to cover all bases, or buses).

The Altair 8800b comes with no memory, so you'll have to buy some before you can start programming. As for other add-on options, there are more for the 8800b (and its compatible brothers) than for any other hobby computer—over 100 boards, from several dozen manufacturers, that can be plugged into the Altair S-100 bus, including memory, graphics, voice synthesizer, peripheral interfaces, etc.

Programs can be entered and run on the Altair 8800b without the need for anything other than the computer itself, although the hard way. You can enter a program, in machine language, byte by byte, by means of the front-panel switches, run the program, and then read the results by decoding the pattern of lights. However, this is suitable only for very short programs, and for occasions such as checking out the computer when you've just built it, or when you have problems with it later on. So unless you enjoy flipping toggle switches hour after hour, you'll need some form of input/output, such as a keyboard for input and a TV set (or other type of video screen) or a printer.

Other Type C microcomputers, in addition to those already mentioned, include the PCM-12A Electronic Tool Company ETC-1000, and the MITS Altair 680b.

All in all, with what you'll have to invest for the computer and for input/output, this is the type of machine to choose if you're sure you want to go further than just becoming familiar with computers, and if you're more interested in programming than in hardware. Also, it's the type to choose if peripherals are your thing, since so many are available for the Altair S-100 bus.

Type D: Box Without Lights or Switches

Are all those switches and lights really necessary? Not really, since there are very few microcomputer operations that can't be performed using a keyboard, if the computer has a monitor. This is a collection of relatively short service programs, stored in a read-only memory (ROM), that can greatly shorten the time required to get your programs written, debugged, and running.

One of the most important of these service routines is a bootstrap loader, which has to be fed to any microcomputer before anything else goes in, to guide the regular programs to the proper places in memory. Since the computer is a very fast but stupid servant, it has no way of knowing just where your programs should be loaded in memory, so the bootstrap loader is needed as a set of signposts.

Other monitor programs can include, as in the Southwest Technical 6800 computer, which is the best known of this type, routines for examining (and changing, if required) the contents of any memory location, printing or punching the contents of any memory location, displaying the contents of the MPU registers, and switching the computer over to running the user's program.

Type D: Southwest Technical 6800

Other computers of this type include the OSI Challenger, PolyMorphic Poly 88, Processor Technology Sol, and the Wave Mate Jupiter II. Most of these computers have only one or two switches on the front, usually one for power on/off, and another for reset. Just about the only function one of these computers can't perform, and which a Type C can, is to sense the position of a front-panel switch. That is, the program can ask, in effect, "Is switch 3 up or down?" in cases where it's easier to set one or more switches on the front panel than to have to change the program each time there's a difference in the conditions represented by these switches.

Just to confuse things, there's at least one computer, the MITS Altair 680b, which has both a full set of switches and lights, *and* a monitor in ROM. Incidentally, nearly all the Type B all-on-one-board computers contain a monitor.

The Type D computer is the way to go if you want a faster startup and the other advantages provided by a monitor. The monitor puts the bootstrap loader into memory as soon as the computer is turned on, or whenever you press the reset button, so you're ready to load your own programs. With a Type C computer, you'll have to key in the bootstrap loader by hand each time you turn on the machine, although this doesn't take too long once you've gone through it several times.

Because the Type D machine has only a couple of switches, it's somewhat cheaper to buy, on the average, and easier to put together; the Southwest

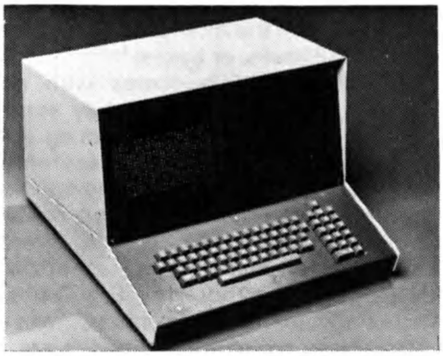
Technical 6800 is admired even by competing manufacturers for its ease of assembly.

Type E: All in One Box

With this all-inclusive type of computer, you get everything you need all at once, all in one box: computer, crt, and keyboard. Although this type is the most expensive, since it combines a computer and two peripherals, it does provide a complete package that lets you start programming right away, because most of these come with a monitor.

Computers of this type include the Compucolor 8001 and the Sphere 300 series.

This type of computer is important, because, in prewired form, it will probably become the most common "people's computer" of the future. More and more people who can't wire up a kit are getting interested in hobby computers. So this type is bound to be a best seller, since it comes complete and ready to use, with no add-ons needed unless the user wants more memory. A forerunner is scheduled to be on the market very soon: the PET 2001, made by Commodore (which owns MOS Technology, creators of KIM-1), with molded plastic case, keyboard with graphics characters, numeric keypad, built-in 9-inch video monitor, 20-line by 40-character screen, built-in audio cassette unit, 4K RAM, about 12K of ROM with BASIC interpreter and cassette operating system, all for $495. Production was scheduled to start July first of this year.


Type E: Sphere 310

Several of the larger hobby-computer kit manufacturers are considering wired-only BASIC models, at least two calculator manufacturers are working on prototypes of such machines, and even some of the business-computer makers are keeping a close watch on the hobby market to see if and when they should get into this fast-growing field.

L'Envoi

There you have the five main types of hobby computers. There are half a dozen other types, represented by one or two companies each, such as the computer built into an attache case.

Even with the guidelines we've provided, making a choice still won't be a simple matter. Write to manufacturers for information. If you can visit a computer store, let the personnel demonstrate their products, and they'll be happy to answer your questions. Join a computer club, and talk with hobbyists who have operating systems and whose experience can help you choose a machine that's best suited to your wants and/or needs.

Magazines and club newsletters are excellent sources of information—take a look at *Creative Computing's* "Equipment Profiles." If you can get to a hobby-computer convention, such as held in New Jersey (Trenton and Atlantic City), Cleveland, San Francisco, Atlanta, and several other cities, these are great places to check out dozens of computers and peripherals in a single day, as well as listen to lectures on hardware, software and applications.

Don't wait. Get in on the fun now. For as little as a few hundred dollars (or for as much as a few thousand), you too can become a micro maven. ■

★ ★ ★ ★ ★ ★ ★ ★ ★ ★ ★ ★ ★ ★ ★

Heath: Two Computers and Two Peripherals, for Openers

Stephen B. Gray

Heath Company, at last silencing the rumors that have been flitting around for several years, introduced their line of "personal computing products" at a June 1 press party in Benton Harbor. First public showing is scheduled for the Personal Computing Fair in Atlantic City, NJ, August 27-28. Four products are due to be available this Fall:

H8 8-Bit Computer

At the lower end of the line is the H8 8-bit computer, using the 8080A MPU. An "intelligent front panel" features a 16-button keyboard for octal data entry, a nine-digit segmented octal display (six for address, three for register or memory contents). and four LED status lights, providing what Heath says is "the most controllable computer on the market."

The bus, which will probably become known as the Heath or S-50 bus, uses 50-pin connectors on a ten-slot motherboard.

A built-in 1K ROM monitor controls the front panel and the load/dump

H8 8-bit computer

operations, and permits turnkey operation. A built-in programmable speaker provides special effects as well as feedback signals to indicate proper or improper operation: a short beep means, for example, that your keyboard entry was correct; a long beep would mean that you made some mistake in the entry.

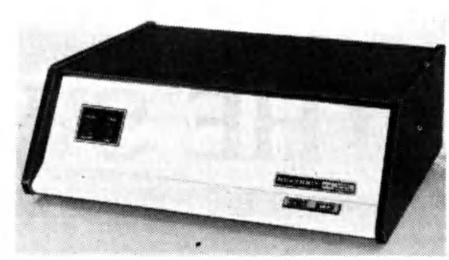
The CPU is wired and tested "for maximum success," on the theory that if your CPU board doesn't work, you can't check out anything else. All other H8 boards are in kit form, and all ICs are socketed.

Mail-order price of the H8 (without memory) is $375, including all systems software on audio cassettes. An 8K memory board with 4K of static RAM is $140, a 4K expansion chip set is $95. The H8 cabinet is configured for 32K of memory; the H8 can address 65K of memory. The serial I/O interface board with 1200-baud audio-cassette interface is $110; a three-port parallel interface is $150. Parallel and serial ports are software-compatible; with the 8251 UART, the software doesn't know if it's talking to a serial or parallel port. All software works with any configuration; no program changes need be made if different peripherals are used. The serial interface uses the Kansas City standard tones and self-clocking for cassette recording, and also has complete modem control.

H8 software includes BH BASIC (Benton Harbor, 8K), Extended BH BASIC (12K), text editor, two-pass assembler, debug, and panel monitor. An interesting feature is command completion: for instance, if you type PR, the computer completes the word by typing INT. Error detection during tape handling is provided. Both BASICs support PEEK/POKE, PIN/OUT, and the SIN, COS and LOG functions are said to be one and a half times faster than in anybody else's BASIC (making for a faster Startrek game). To demonstrate the H8's multitasking capability, one game (Chase) was played on the display, and another (Hangman) on the CRT.

H11 16-Bit Computer

At the top of the line is the H11, using the Digital Equipment Corporation LSI-11 Microcomputer Module. Again, the CPU board is fully wired and tested; all other H11 boards are in kit form, including 4K static RAM at $275, serial interface ($95) and parallel interface ($95). The CPU board will accept a hardware multiply IC, six boards can be plugged into the backplane, and memory is expandable to 20K.



H11 16-bit computer

The H11, at $1295 mail-order for the CPU and 4K RAM, includes a complete DEC system software package, containing editor, PAL-11 assembler, linker, online debug package, input/output executive, BASIC (8K) and FOCAL (both 4K and 8K). Purchasers of the H11 will be eligible to join DECUS, the DEC Users Society, "which has a library with over 800 programs designed for the PDP-11 family of computers, many of which were developed for or can run on the LSI-11." The H11 BASIC is said to be 2.5 times faster than any other BASIC (a future development will make it ten times faster), and features strings, PRINT USING, and deletable features.

H9 CRT Terminal

The H9 alphanumeric video terminal has a 12-inch screen, 67-key ASCII keyboard (upper case only), 12-line by 8-character format (with format option of four columns of 12 lines by 20 characters), cursor control, batch transmit, and a plot mode for very simple graphics. Baud rate is selectable, from 110 to 9600; standard serial interfaces include EIA, 20-ma loop, and TTL input/output.

The H9 terminal was demonstrated with the H8 computer, although it can be used with the H11 or any other computer. To load, the user puts a cassette in a cassette player/recorder (Heath will provide a GE unit for $60), presses the LOAD key, and when the program is fully loaded, a short beep is heard. The Timing and Processing board is pre-wired and tested; the rest of the terminal is in kit form, at $503 mail-order.

H9 CRT terminal

H10 Paper Tape Reader/Punch

Designed for use with Heath's H8 and H11 computers, the H10 paper-tape reader/punch will work with any other digital computer. The tape is standard one-inch roll or fan-fold eight-level tape, read by the H10 at 50 cps and punched at 10 cps. Independent punch and reader circuits permit simultaneous operation, as in copy mode for tape duplication. Mail-order kit price for the H10 is $350.

Support

Heath is going all-out to back up their computer products, which represent the largest investment in the company's history. There will be self-instructional programming courses, and a Heath Users Group (HUG) with a newsletter. The group that provides technical help to kit builders over the telephone is gearing up for an expectedly heavy load of calls on the computer products.

A $150,000 training program is aimed at providing at least one technician in each of the 60 Heath centers with the know-how required to service the computer line.

Over ten man-years were spend in writing the various mnuals or the computer line, and over 600 pages of software documentation are in preparation.

The operating manuals will contain step-by-step instructions on what to do with the keyboards and displays, and will show memory maps, I/O port maps, bus designations, etc. The software manuals will contain many examples of program use, and many complete listings, such as the entire floating-point program in BASIC.

H10 paper-tape reader/punch

Other Items

Heath will make the LA36 DEC Write. available although the price hasn't been set yet for this 30-cps teleprinter with 132-column format.

Future plans include floppy disk, more software and interfaces, prototyping cards, graphics, and wired versions of the computer products. —SBG ■

BUILDING THE SWTPC 6800

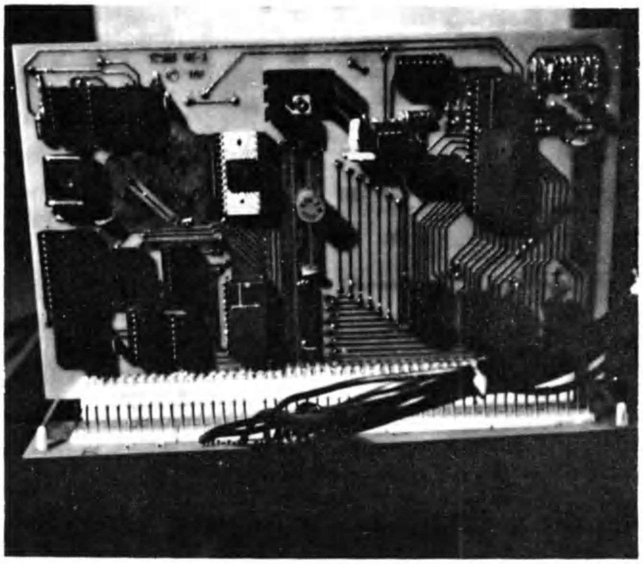
by
Bryan Loofbourrow
P.O. Box 1237
Mountainside, N.J. 07092

After reading the article by Steve Gray[1] about building an Altair 8800, I was surprised at the comparative ease with which my kit went together. Although the SWTPC 6800* is not without its share of construction hassles, I found that the problems mentioned by Steve of wiring in panel switches, over-complexity of programming manual, and memory bit errors did not exist in the SWTPC machine.(In all fairness, it should be pointed out that the Altair 8800 Steve built was one of the very original ones from MITS.—DHA

Like the Altair 8800, the SWTPC 6800 has building instructions with schematics, PC board layouts and component placement drawings. One source of potential trouble, especially to inexperienced kit builders, is the set of instructions regarding installation of resistors and capacitors. For example, the directions suggest that the builder mount all of the resistors first, installing each one "from the TOP side of the board bending the leads along the BOTTOM side of the board and trimming so that 1/16" to 1/8" of wire remains"[2]. I think that there are two things wrong with this. First, if you install all of the resistors on a given board then go back and solder all of them at once, you stand a good chance of missing a couple of connections. Second, bending the leads along the board and *then* cutting makes it almost impossible to cut the leads without digging into the board, and wasting a lot of effort being careful of the board in the process. These may be minor issues, but I think I saved myself a lot of effort and potential trouble by soldering each resistor right after installation, and leaving the leads straight while I soldered, cutting each lead after the connection was made.

To avoid solder bridges a good eye, a steady hand, and a small soldering iron are essential.

There are five boards that come with the computer: mother board, serial control interface, microprocessor board, power supply regulator board and memory board. Although special care required in building the microprocessor board can make work there rather tedious, by far the most difficult board to assemble is the memory board. The reason for this is that the circuit design required that conductors run in between IC pins, and, as a result, the pads are very thin and very close to adjacent connections. To avoid solder bridges a good eye, a steady hand, and a small soldering iron are essential when working with this board. If any bridges are formed, however, they are fairly easily located through close inspection with a magnifying glass and by using the memory diagnostic program supplied in the system documentation notebook.



Heart of the computer is the MP-A microprocessor board. Notice the use of IC sockets (which are not included in the kit).

IC sockets are not supplied with the kit, except for a couple of the most expensive IC's. While you may want to buy and install your own sockets (I did), SWTPC advises against it. They say that sockets are just one more thing to go wrong. I say that I haven't ever had a bum socket and if I ever want to remove an IC for repair or modification of the board, I would much rather be able to unplug it than to have to go through the trouble of desoldering it. This issue is, of course, something for the individual builder to decide.

The documentation supplied with the computer is extensive; it includes a programming manual (194 pg.) and a notebook full of IC specs, explanations of stacks, interrupts, etc., several diagnostic programs and a TIC-TAC-TOE game program. All of the programs in the notebook will run in the 2K of memory supplied. Much of the manual is of use only to the knowledgable, but can nevertheless be invaluable when making hardware modifications.

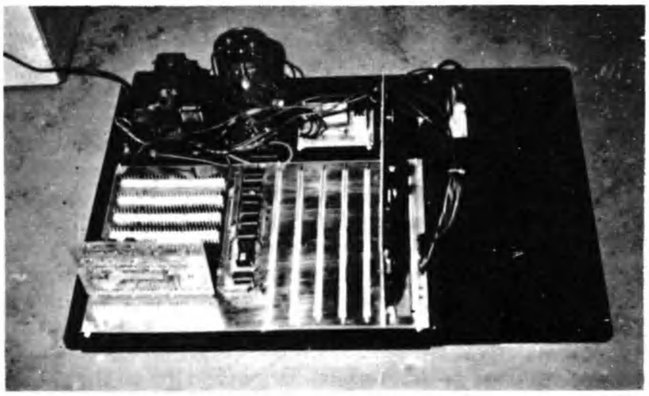
Perhaps SWTPC's biggest point of pride is the MIKBUG monitor program, which enables the user to connect his terminal (RS-232 or TTY) to the serial control interface and immediately be able to communicate with the computer. MIKBUG itself is on a ROM (Read Only Memory) which retains its memory even with power off. The program is automatically loaded and run when power is applied to the system, or when the user presses the RESET switch, one of the only two switches on the front panel of the computer (the other is POWER ON/OFF). The MIKBUG program occupies 512 bytes of memory and has the following functions: Memory Change/Examine (M), Memory Dump (P), Load Program (L), Run Program (G) and Register Printout (R). These functions are essential when testing and debugging programs, and serve to simplify the use of the system greatly.

---

*Southwest Technical Products Corp., 219 W. Rhapsody, San Antonio, TX 78216.

Software that has been written for the 6800 includes four versions of BASIC ranging from Tiny BASIC to an extended 8k BASIC as well as several games and specialized subroutines. About twenty different programs are available through an ad supplied with the kit.

SWTPC has greatly simplified the construction and use of their computer as compared with some others.



SWTPC 6800 inside view. The front panel is face down at the right.

Construction time for the system is about 14-16 hours; cost is $395 (low, as microcomputers go). I think that SWTPC has greatly simplified the construction and use of their computer over some other models; although I don't know how their 6800 compares with the ALTAIR 680. Nonetheless, "second thought" modifications are virtually non-existent in their kit, and the instructions are reasonably clear to anyone with a minimum of kitbuilding experience.

REFERENCES

1. "Building a MITS ALTAIR 8800," *Creative Computing,* Jan.-Feb. '76, p. 13.
2. Taken directly from the MP-A building instructions.

How I Built an IMSAI 8080 With Solder, Luck, and Very Little Help From the Manual

by Steve North
Newfoundland, New Jersey

It's interesting to consider that amateur computer kit building took off only a few years ago with the introduction of the Altair 8800, and now it's possible to buy not only Altair-compatible boards from second sources, but also entire systems which resemble the Altair but hopefully feature some design improvements. One of these is the IMSAI 8080. After looking at ads that read, "Altair Good, IMSAI Better!" for a few months (I couldn't decide if they were trying to conserve words or if the ad was written by a Japanese ad agency) I finally succumbed and bought an IMSAI through a local computer store.

A basic, no-frills IMSAI consists of a CPU card, memory, front panel, power supply, motherboard, and chassis. The physical construction of the IMSAI seems simpler than that of other kits such as a Heathkit stereo, or, ahem, the IMSAI's very popular competitor. It takes about 40 hours to assemble an IMSAI if you do a slow, careful job. Most of this time is spent soldering parts to printed circuit boards.

The motherboard, into which the other boards in the system are plugged, runs from front to back on the left-hand side of the cabinet. The power supply is located on the right side of the cabinet. All the power supply components including a very hefty transformer and the filtering capacitors are mounted on a printed circuit board. (Apparently the power supply in my IMSAI is a recent revision because IMSAI is still showing the old power supply in their advertisements.)

To connect the front panel to the system bus, IMSAI just runs the motherboard out under the front frame and so the front panel plugs into the motherboard and a small ribbon cable connects the front panel to the CPU. A "sandwich" consisting of layers of plastic, paper, and a photographic mask, is blotted with spacers to the front panel and bears the legends for the switches and LEDs. No jumpers are required on any of the PC boards.

So, a kit like this should be a snap to build, right? Well, not quite. It seems to be understood these days that someone building a computer kit should have typical kit-building skills — ability to solder, identify parts, and follow directions. Nevertheless IMSAI doesn't make things too easy for you. For instance, after assembling the CPU board according to the instructions, you have an inductor and a resistor left over, which weren't mentioned in the instructions. Since the inductor appears in the assembly diagram and in the photograph of the finished board, it's logical to assume that IMSAI just forgot to mention it in the instructions. That leaves you with a resistor, which *is* in the assembly diagram, and *is* in the schematic of the CPU board, but which *isn't* in the photograph of the finished board. I installed it anyway.

There are other major slipups in the manual. The instructions describing installation of the front panel switches were hopeless, but the process is simple enough to figure out from studying the diagrams and photographs. In the section on the construction of the power supply, you're told to bend the leads of some rectifier diodes so that they will stand upright on the board, cathode down, and then install them as shown in a diagram. But if you install the diodes as shown, two will be in backwards! A quick comparison with the schematic and photographs reveals the correct installation. How about that! A manual that helps you blow up your computer! Another less serious blunder in the instructions is the labeling of the filtering capacitors "C1-C4" in the instructions, and "CO-C3" in the assembly diagram.

The first page of the manual lists in red ink all the things not to do, if you don't want your warranty voided. One of these items reads, "Do *not* leave out any construction step, including taping of the power traces." That's funny, because nowhere in the manual did it say anything about taping power traces. Hopefully this instruction is merely a reference to the old power supply. If not, readers desiring to know how it feels to void your IMSAI warranty in front of thousands of people can write to Steve North, c/o *Creative.*

I never did figure out how IMSAI expected you to cut little holes, ⅛" diameter, in the photographic mask that goes over the front panel. It's pretty hard to do with a kitchen knife or a pair of scissors! IMSAI really should trim the mask for its customers, since it is a little difficult and it is important to the appearance of the finished product.

Finally I had the IMSAI finished and turned it on. Would it work? Of course not! The LEDs that indicated what was happening on the data bus stayed on all the time, and did strange things when the system ground was touched. The address bus LEDs seemed to work nicely and increment on an Examine-Next or Deposit-Next. I had a very expensive 16-bit binary counter!

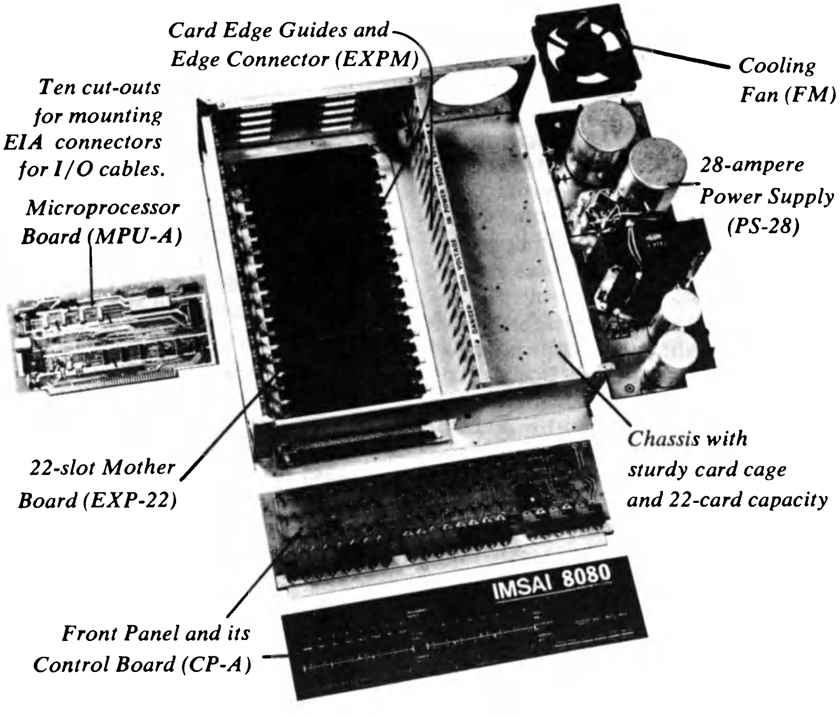
Probably someone reading this right now knows exactly what the problem was. Not being especially good (Read, "Not any good") at debugging I took the system down to my computer store where the owner tried out a newly assembled front panel board in my computer. 'Well,' he said, 'at least there's good news. I have a front panel that works.' I was delighted. Anyway, the dealer advised me to send the board back to IMSAI, which I did. It came back *two months later.* It only took a month to construct the computer! I'll have to order something from IMSAI and take two months to pay.

Once reunited with the wayward front panel, the IMSAI ran perfectly. There are two checkout programs in the manual, explained in great detail. The first reads from the front panel sense switches and displays the data on the programmed output port also on the front panel. A minor variation of this program complements the data before it is displayed. The second program is a game called "Kill the Rotating Bit" which is played with the little blinky-lights and the sense switches.

IMSAI also includes two other books with the system. One of these is Adam Osborne's *An Introduction to Microcomputers,* which discusses the programming of microcomputers in general. The other is the *Intel 8080 Microcomputer Systems User's Manual,* a book of information on the inner workings of the 8080, the 8080 instruction set, and specs on other chips in the series. Apparently IMSAI decided to include these books rather than try to explain how to program their system in their manual. However, the IMSAI manual does contain several pages of information on how each board works and a nice explanation of what the switches and LEDs on the front panel indicate.

The only problem reported with correctly assembled hardware is that noise on the 110 VAC trace running on the front panel to the on-off switch interferes with the operation of the deposit one-shot. Several fixes have been suggested, one of which is the installation of a Pi-filter on the A.C. line, which is already present on my power supply. Now if IMSAI could just bring their documentation and support up to the level of their hardware ....

Despite a manual which was probably written by 100 monkeys pounding on 100 typewriters day and night (actually that's being a little harsh) and the blinding speed of IMSAI's service department, the IMSAI 8080 is still a worthwhile, cleanly designed system. Now if I could just afford 65K of memory! ■


Card Edge Guides and
Edge Connector (EXPM)
Ten cut-outs for mounting EIA connectors for I/O cables.
Microprocessor Board (MPU-A)
Cooling Fan (FM)
28-ampere Power Supply (PS-28)
Chassis with sturdy card cage and 22-card capacity
22-slot Mother Board (EXP-22)
IMSAI 8080
Front Panel and its Control Board (CP-A)

Blow up view of the IMSAI 8080 from the manufacturer's catalog. Available for $1.00 from IMS Associates, Inc., 14860 Wicks Blvd., San Leandro, CA 94577. (415) 483-2093.

A Creative Computing Equipment Profile . . . .

TELETYPE MODEL 43 TERMINAL

by David H. Ahl

From the people who brought the world the slow and noisy, but incredibly versatile ASR 33, we now have, some 600,000 ASR 33's and 10 years later, the fast, quiet, more versatile Model 43.

The Model 43 is initially available in only a KSR version (keyboard send-receive, i.e., no paper tape or local storage). Presumably at some future point, local storage will be an option.

The KSR 43 operates at 10 or 30 characters per second with corresponding transmission speeds of 110 or 300 baud, switch selectable. It prints sideways on 12" wide paper using a very clever 7 x 9 dot matrix to provide 132 character width across the paper in 10". In other words, horizontal character spacing is 13.2 per inch; lines are spaced vertically at 6 per inch. The almost 2 to 1 horizontal to vertical character spacing should allow some very good graphics and plotting. The 7 x 9 dot matrix printer also allows a full 94 ASCII code character set (upper and lower case) and is very legible. The 12" wide by 8½" long pinfeed, fan fold paper can be trimmed to standard 8½ x 11" by removing the perforated margins thus making the output easy to bind or file.

The solid state keyboard has the same layout as an office typewriter so it can be readily used by anyone who can type. A built-in buffer absorbs momentary keyboard bursts. Using control keys, the keyboard can generate all 128 ASCII code combinations.

A clever keyboard feature is a "CAPS LOCK" key that functions similar to the "SHIFT" key, but with one important advantage. When it is locked (depressed), it causes capital letters to be generated without having to operate the SHIFT key but also automatically generates numerics and other unshifted characters.

A mode select key allows half or full duplex operation. There is also a parity on/off key, and an interrupt key. An automatic print test key verifies printer operation by causing all 94 ASCII characters and a special parity error symbol to be printed.

Additional features include last character visibility, on-line back space, on-line margin settings, automatic carriage return/line feed on received data, and continuous line feed.

The KSR 43 has a 103-compatible data set integrated right in the unit with automatic answer capability.

Reliability of the KSR 43 should be excellent. It has built-in self diagnostics, component modularity (no more hours of digging into a messy boar's nest to replace a key contact), and state-of-the-art MOS circuitry.

The KSR 43 is amazingly compact measuring 18" wide x 21" deep x 5½" high and weighing in at 30 lbs. The housing is an attractive off-white and charcoal grey. Optional accessories include pedestal and copyholder.

Major entries in the medium-speed ASCII printing terminal field include the DEC LA-36 DECwriter II, GE Terminet 30, TI Silent 700 and a new entry from Data General. It appears that Teletype has put together a package in the Model 43 that will be a real winner in this competitive field.

Currently, the KSR 43 is available only from four Bell System Operating Companies but will soon be tariffed in all companies. A variety of pricing options are available ranging from straight rental (approx. $90 per mo.) to various two-tier (front-loaded) plans. As of this writing there are no announced plans for outright sale of the unit by Teletype Corp.; for now you'll have to turn to your local phone company. ■

The Sol-20: Simple Enough For a Six-Year Old

by Steve North

One of the most obvious trends in personal computing is toward the "complete" computer kit — one which contains all or most of the ingredients of a usable computer system. Increasingly, units are being offered assembled too. The "complete" unit not only makes putting together a system easier (since the buyer need only select a single package, rather than a mainframe from one manufacturer, memory from another, etc.) but also makes the use of a system easier since it often features a monitor on PROM. An outstanding (but by no means the only) example of this kind of computer is Processor Technology's SOL System.

The foundation of the SOL ps a single large PC board containing:

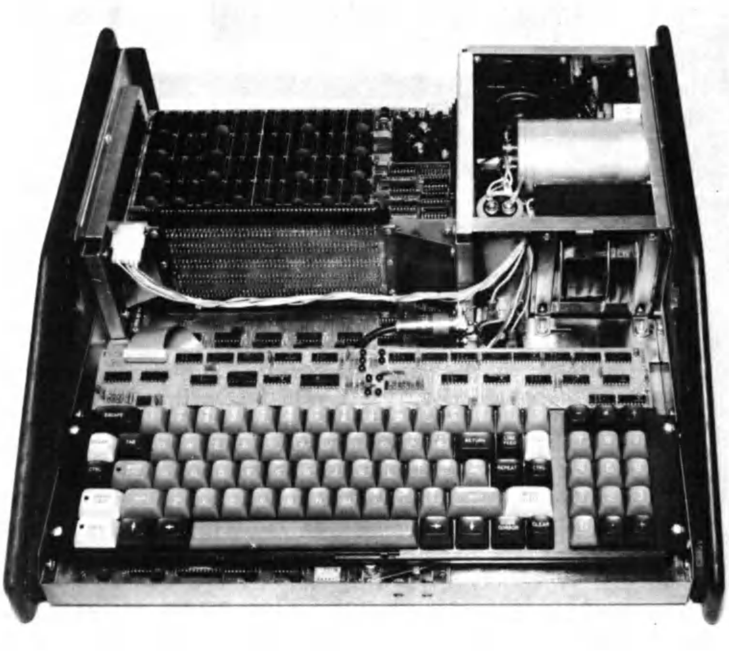
- An 8080-based CPU. Let's not dredge up all the old 8080 vs. other MPU's argument; suffice to say that the 8080 *is* one of the most popular MPU's in amateur computing and there is a large body of support software for it (not to mention some interesting hardware).
- A Personality Module. That's PTCO's fancy name for a small (1½" x 3") PROM card which plugs into the SOL, containing a hex monitor. Having a monitor on PROM eliminates the need for an expensive and sometimes unneededfront panel. The PROM card is not designed for the S-100 bus, hence it keeps the price of the system down while maintaining a level of flexibility which would be lost were the PROM built into the main PC board.
- 1K RAM, intended primarily as scratch pad for the Personality Module.
- The equivalent of a Processor Technology VDM-1. For those of you not familiar with amateur computing hardware, a VDM (Video Display Module) is a high speed video driver, which, with the proper software, can be made to simulate a fancy CRT. If you don't need hard copy, it's all you need for humanreadable output. You will need a TV to connect to it. Output for a video monitor or modified TV is provided, however, it is a simple enough job to mount a Pixeverter inside the SOL for direct RF entry into a TV set.
- A parallel and a serial data port. These permit you to use peripherals such as Teletypes or optical papertape readers with your SOL without buying separate interface boards.
- Two cassette interfaces with motor control. They operate with both the Byte/Kansas City Standard, and PT's new 1200 baud standard (ever wonder why they're called standards anymore?). Sophisticated use of cassettes for mass storage in the future will no doubt require computer control of the cassette motor and perhaps two tape units — at least.

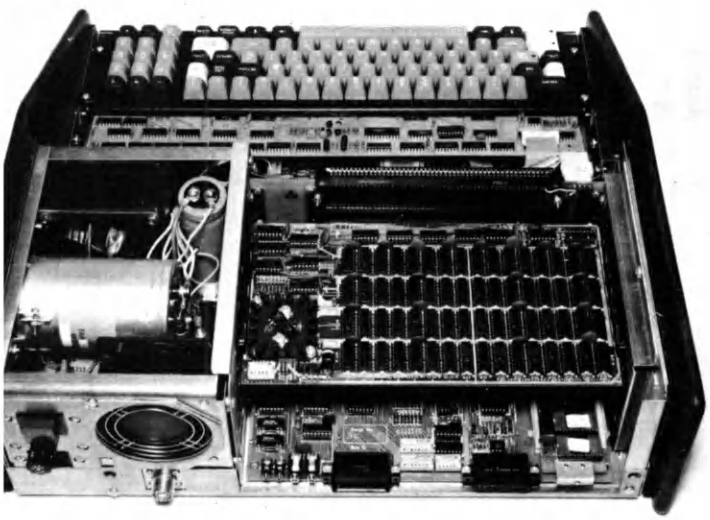
The SOL also includes an 80-key keyboard with the full ASCII character set, as well as special function keys such as SHIFT LOCK, UPPER CASE, LOCAL, MODE SELECT, et al. The SOL power supply seems adequate for the job. There are also five S-100 compatible slots in the SOL. Offhand that doesn't seem like a lot, because most of your memory has to go here. Still, five 8K RAMs gives you 40K of memory which is more than most people have. Processor Tech is already marketing a 16K RAM board, and you can be sure a 64K RAM board isn't too far away. If you plan major expansion of your computer, remember that a floppy disc interface, TV Dazzler, and music-making module would only leave one slot in the SOL for memory. That's the price you pay for compactness. But if you don't plan to make your computer a continuing investment or put every application in the book on it, the SOL should be all you need. The entire unit is housed in a nice-looking cabinet with walnut sides, about the size of a portable electric typewriter.

We'll leave details on building the SOL to other hardware-oriented magazines. However, construction of the SOL looked fairly typical for a computer kit. It did seem that there were a lot of "Engineering Modifications" because of errors or updates in PC design, requiring cutting of traces and running jumper wires. Of course, it would be logical to expect Processor Tech to revise the boards so future buyers may not have this problem.

When the SOL is turned on, the program in the Personality Module initializes the system and enters the *terminal mode*. In this mode, what is typed on the keyboard is sent to the serial port and data received at the serial port is displayed by the Personality Module Software via the VDM to your TV set. The I/O at the serial port is either half or full duplex, selected by a switch inside the SOL. So for the price of a SOL (and a TV set) you have a nifty little terminal.

Sol-20 with covers removed. Front (or keyboard) is in foreground, power supply is in right rear corner, expansion chassis (with 8KRA Memory installed) is to left of power supply. The vertical board just behind white connector on left is the backplane board.





Sol-20 with covers removed. Rear side of assembly is in foreground and Sol-PC is just visible at lower right rear of assembly. 8KRA Memory is installed in expansion chassis above Sol-PC.

But SOL is also a *stand-alone computer.* Hit mode-select, and you're running the hex monitor located on the Personality Module. Three versions are available — CONSOL, SOLED, and SOLOS. CONSOL, the simplest, permits you to enter and dump memory in hex, execute a user program, return to terminal mode, or to load a program from cassette tape. The BA (for BASIC) command executes a user program at 0000. This suggests that a person could just turn on the SOL, hit mode-select, type TL to load BASIC, then type BASIC and go, without knowing anything about machine code.

Indeed, with SOLOS, one has only to type TXEQ BASIC/1, and you're off and running (see box). SOLOS provides more sophisticated I/O handling and tape cassette commands than CONSOL, while SOLED is designed for advanced editing features.

These monitors permit dynamic assignment of the input and output devices to be used. Thus, user programs can use the monitor for their I/O operations and you can change I/O devices without patching the program. The default devices are the keyboard for input and VDM for output. Most of our experience has been with the SOLOS monitor which we've found exceptionally easy to use. More important, it's had the flexibility to do virtually anything we wanted.

One thing we liked about the CUTS (Computer Users Tape System) cassette was that Processor Tech has standardized the format of the data to be used on their object tapes as well as the actual means used to record it. The format includes a header label with information on the name of the file, executing address, and length. That may not seem like a big deal, but if your system merely saves one huge block of data on a tape and then a checksum (a la Tarbell and others), it is impossible to search for a particular file, or even find out what a file is. On the other hand, once people are using a simple (standard?) format for exchanging data it's difficult to get them to change. Also, it takes more software to process sophisticated data formats and nobody we know likes to toggle in long tape handling routines, however, if you have a nice SOLOS monitor in PROM, who cares?

I guess what we're saying is that the SOLOS monitor and CUTS cassette system is great for saving and retrieving programs for your own use. However, this combination is not likely to be adopted as an industry standard, hence you'll probably be limited to exchanging programs with other SOL users. In most cases, this is probably not a disadvantage, but just a factor to be considered.

Processor Tech supplies an expanded version of their 5K BASIC with the SOL. It is a fairly typical BASIC — floating point math, one-dimensional arrays, multiple statements per line, etc. One very handy feature — it permits writing and reading data from a CUTS cassette. Unfortunately, 5K BASIC seems a bit klugy and sometimes limiting. For instance, to prematurely exit a FOR/NEXT/loop, you must set a switch and finish the loop, *then* branch. There have been complaints about formatted PRINT statements not working properly. In our own rather extensive use of 5K BASIC in the past few months we've found rather unexpected ways to restart the interpreter with an arithmetic expression, and to crash the interpreter with an undimensioned array or with a peculiar FOR/NEXT loop combination. Granted, this is abuse of the interpreter but it was found accidentally, not intentionally, and it must be expected that other people will do the same.

Despite the minor criticism of 5K BASIC, the system is eminently useful. A number of CAI programs (described elsewhere in this issue) were written on the system. But perhaps the best testimony comes from 3 children, ages 6, 7, and 8 who simply follow a set of instructions (see box) *completely on their own* to run their programs. It's difficult to find another system — mini, micro, or timesharing terminal — that's this easy and straightforward to use. ■

Complete start up and shut down instructions for a Sol 20. The system is used regularly by 6,7, and 8 year old children for math drill and practice.

STARTING UP

1. Turn on computer and TV set, upper case should be lit
2. Plug in cassette recorder
3. If tape is not rewound,
    A. Press "REW" on recorder
    B. Type TC↵ (↵ =Return Key)
    C. When tape is rewound, press "MODE SELECT"

Tape Counter

4. Press "PLAY" on recorder
5. Type TXEQ BASIC/1↵ 0-25
    Screen should say: Sol BASIC-5
    READY
6. Type XEQ-CAI↵ 25-28
    (Addition drill and practice)
7. Press "MODE SELECT" To exit program

6a. Type XEQ-MULTI↵ 28-32
    (Multiplication and division)
7a. Press "MODE SELECT" to exit

6b. Type XEQ-GUESS↵ 32-35
    (Guessing game)
7b. Press "MODE SELECT" to exit

To restart a program, Type "RUN" key
To clear an error, Type "DEL" key

SHUTTING DOWN

1. Press "STOP" on recorder
2. Turn off computer and TV set
3. Unplug recorder

Radio Shack's $600 Home Computer

Wes Thomas

This September you can walk into any of the 140 major Radio Shack stores around the country and buy a fully wired and tested TRS-80 microcomputer for $599.95. For that amount, you get a compact ASCII keyboard with built-in microcomputer, plus a video monitor, and an audio-cassette recorder to store programs or data. You can take all of these home, plug them in, and write your first program in Radio Shack's "Level I" BASIC. Or you may decide to buy just the microcomputer/keyboard for $399.95, and use your own TV (through an RF modulator) along with your own home audio-cassette recorder.

Radio Shack has thoughtfully provided five groups of programs for the TRS-80 on audio cassettes: "Blackjack and Backgammon" (free), "Payroll" (for up to 15 people, at $19.95), Educator—Math I" ($19.95), "Kitchen" (menus, conversion tables, computer directory and message center, for $4.95), and "Personal Finance" ($14.95).

I had a chance to try out the TRS-80 at Radio Shack's press conference in New York on August 3. I found it easy to use, with a compact, comfortable keyboard, a legible 12-inch video display, and a well-written, thorough, and very patient instruction manual. The video display features automatic scrolling, and displays 16 lines of 64 characters for text (software-selectable to 32 characters per line).

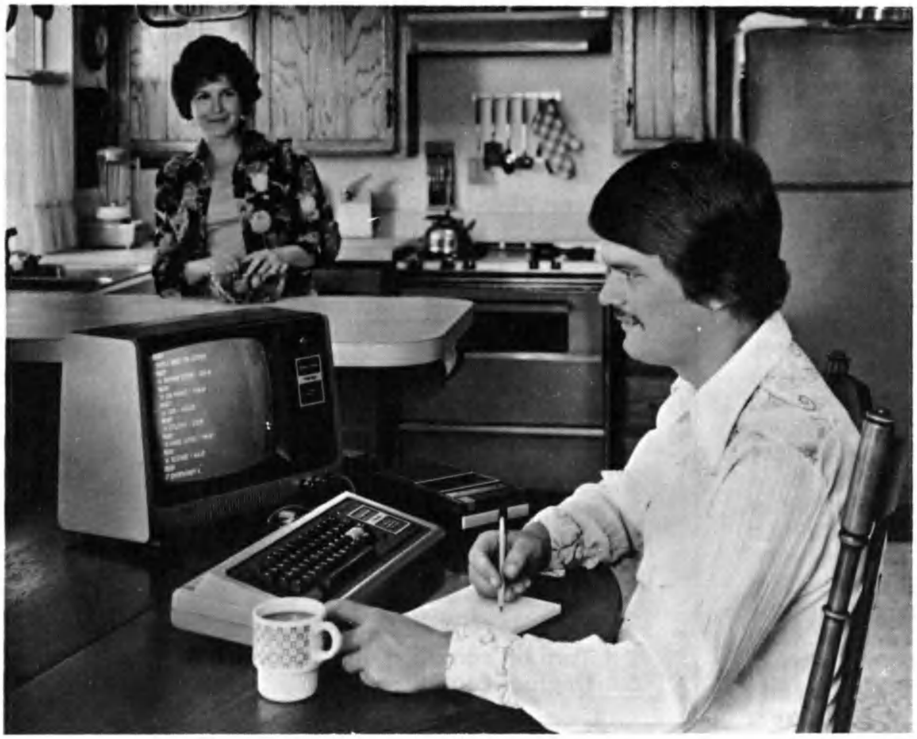
Father of the TRS-80

The TRS-80 is the brainchild of 25-year-old Steve Leininger, who spent two years designing National Semiconductor's SC/MP Development System. Steve was originally hired by Radio Shack last year to develop a computer kit, but, having worked in a Byte Shop in California, he quickly convinced Radio Shack that "too many people can't solder." Steve says he put together the TRS-80 based on software ideas from *Creative Computing* and other sources.

Enhanced Tiny Basic

The TRS-80 uses Radio Shack's "Level I" enhanced Tiny BASIC interpreter, which resides in 4K of ROM (Read-Only Memory). Level I improves on Tiny BASIC by adding floating point (to provide the decimal point), and scientific notation. And it adds graphics commands that let you easily generate pictures, which you can mix in with words. The storage capacity for programs plus data is 4K (about 500 characters) in dynamic RAM (read and write memory). When you run out of space in internal computer memory, you can use the cassette recorder to extend it. (Note to hardware fans: Radio Shack is using the Z-80 chip for the TRS-80, but not the S-100 bus. And they're using "Radio Shack" cassette-tape protocol, which adds one more to the dozen tape formats already in existence.)

Incidentally, the microcomputer is designed and manufactured by Radio Shack in the U.S., the video monitor is from Taiwan, and the cassette recorder is Japanese.



Peripherals

Is that all there is? Nope. Radio Shack has bigger plans for December. Here is what you can get for Christmas, if you can convince your wife (or husband or whoever) that you desperately need a computer to make

your first million (Radio Shack says they will pay for useful programs):

- A compact 80K floppy-disk unit for $600-$700 (including disk operating system).
- A dot-matrix printer for under $1,500 (a $700-or-less printer is in the works for 1978).
- Extended (8K) BASIC, or "Level II," as Radio Shack calls it (this includes extended string capability, and other enhancements that will allow more sophisticated business and educational programming).
- Level III BASIC (with disc-control commands, and assembly-language subroutines). An internal 12K ROM will be available for this, allowing 4K for other uses.
- Assembler (resident in 16K of disk or on cassette). This will permit faster graphics and I/O routines and let you do a lot of fancy tricks.
- FORTRAN IV (in 16K of disc).
- New software packages for small business and education, including general ledger, accounts receivable, inventory control, music theory, long division, algebra, and other programmed-instruction packages now being developed by noted computer-aided-instruction experts.
- Text editor (using disk and 8K of RAM).

Radio Shack is also planning a modem (non RS-232) so you can connect your computer to other home computers or to a time-sharing service and let it talk its own language. Or you can use it like an intelligent terminal for business-data entry. Or send "electronic mail" messages. Or whatever else you have in mind.

Also in the planning stages is an "Expansion Unit" to hold extra RAM memory, up to a total of 62K. Or you can put up to 16K of that inside the keyboard case (the TRS-80 with 16K of RAM is an extra $289) and another 16K inside the disk unit (ROM memory can be expanded in the keyboard case up to 12K). Now you're ready to go into the business of serving small business — and to make your first million.

Future Plans

Radio Shack is also thinking about other features, and intends to provide them if enough customers are interested, such as:

- Color graphics and an expanded graphics instruction set, plus an RF modulator (pending FCC approval) so you can hook up your color TV and invent your own interactive video games and impress your friends.
- An External Device Controller to interface analog signals, such as from a joystick for video games, or from sensors for fire detection, etc. (a convenient 40-pin plug on is located the back of the keyboard unit).
- Music and speech synthesizer, and voice-recognition devices.

Markets

Radio Shack is going into the computer business in a big way. Eventually, 2100 stores will carry the TRS-80. The main target market is small businesses, then education, followed by the consumer market. Radio Shack's principal objective, according to their financial consultant Garland Asher, is to "influence other product lines — like digital scanners — to improve Radio Shack's image, and to move the corporation into higher-ticket merchandise."

Meanwhile, Radio Shack's parent, Tandy Corp., is planning a chain of "Tandy Computer" retail stores across the country; the first one will open Oct. 1 in Fort Worth. According to VP John Gatliff, "We'll carry a broad line of other vendors' items too, and we'll also sell by mail order, starting in October. Our markets are business, education, and hobbyists."

As Radio Shack president Lewis Kornfeld put it, "This device is inevitably in the future of everyone in the civilized world — in some way — now and for as far ahead as one can think."

Quick Comparison with Heathkit H8 and PET Computers

How does the TRS-80 compare with two other recently-introduced low-cost home computers?

- *Heath H8 kit:* $375. Adding the video-terminal kit plus cassette interface plus 8K memory brings the cost up to $1,110.
- *Commodore PET:* About the same price ($595) as the Radio Shack TRS-80. Compact and with 8K BASIC. Drawbacks: a small calculator-style keyboard, small 9-inch video screen, less-powerful MPU chip (6502), and uses the IEEE bus (which requires intelligent peripherals). ■

From the Log of the Mark V Home Computer & Intruder Alarum

```
DANGER! DANGER! DANGER!

I, THE MARK V HOME COMPUTER/INTRUDER ALARUM DETECT A SUSPICIOUS
LOOKING INDIVIDUAL ATTEMPTING TO FORCE ENTRY TO THE BUILDING.
COMMENCING INTRUSION LOG:
    18:33:47.023 FRONT DOOR INTERLOCK BREACHED
    18:34:54.543 PHOTOCELL CHECKPOINT ALPHA PASSED
    18:36:06.105 WEIGHT OF 97 KILOGRAMS DETECTED ON FIRST STAIRSTEP
    18:37:22.133 INTRUDER ON LANDING; DIGITAL PHOTOGRAPH TAKEN
    18:37:22.354 PHOTOGRAPH STORED AND PROCESSED. ANALYSIS INDICATES
                 WHITE ANGLO SAXON CATHOLIC WITH SCAR ABOVE LEFT EYE
                 AND A SLIGHT LISP.
    18:37:22.665 PHOTOGRAPH DISPATCHED VIA TELSAT TO INTERPOL.
    18:37:22.982 INTERPOL RETURNS ID AS "GREGOR TABRASKII", KNOWN
                 COMPUTER THIEF AND KILOBAD SUBSCRIBER.
    18:37:22.995 ENERGIZE DEFENSE SYSTEM.
    18:37:23.442 TARGET ON FIFTH STEP OF SECOND TIER.
    18:37:23.445 DEFENSE SYSTEM READY.
    18:37:25.045 TARGET AT TOP OF STAIRS. LOCK ON 50 MEGAWATT LASER.
    18:37:25.050 LASER LOCKED ON. QUERY SUPREME COURT ROBO-JUDGE FOR
                 PERMISSION TO TERMINATE LIFE OF INTRUDER.
    18:37:25.052 PERMISSION GRANTED.
    18:37:25.053 ZAP!
    18:37:25.054 SWITCH AIR RECIRCULATION SYSTEM TO HIGH CLEAN.
    18:37:25.100 NOTIFY BUILDING ENGINEERING OFFICE OF NEED TO
                 REPAIR FRONT DOOR AND REAR WALL OVER STAIRS.
    18:37:25:125 RESET ALARUM SYSTEM, RETURN LASER TO STAND-BY.
    18:37:25:143 TERMINATE INTRUSION LOG.

MARK V, READY
```

A Creative Computing Equipment Profile....

SOFTWARE TECHNOLOGY MUSIC SYSTEM

David H. Ahl

The opening page of the Software Technology Music System says, "Perhaps you have a computer and have been wondering: 'What else can I do with it?'

Perhaps you are a musician and would like to investigate computer-generated music.

Perhaps you would like to play a concert."

Well, I am none of these. I have plenty to do with my computer, I am certainly not a musician and, although I would like to play a concert, I hardly have the time to devote to doing so. Some of my friends have occasionally accused me of being a baroque freak because when they come over to the house all they can seem to get on my hi-fi is Bach, Handel, Vivaldi and so on. However, my musical tastes actually run from Baroque to the Beach Boys which encompasses a fair amount of ground in-between. As a result I was intrigued with the flyer about the Software Technology Music System for $24.50.

The price covers a very small piece of hardware (a printed-circuit board about 1½"x6" which plugs into a standard S-100 bus and which mounts all of five different parts, three resistors and two capacitors), an instruction manual, and a cassette tape of the software. It sounded like an intriguing investment for only $24.50.

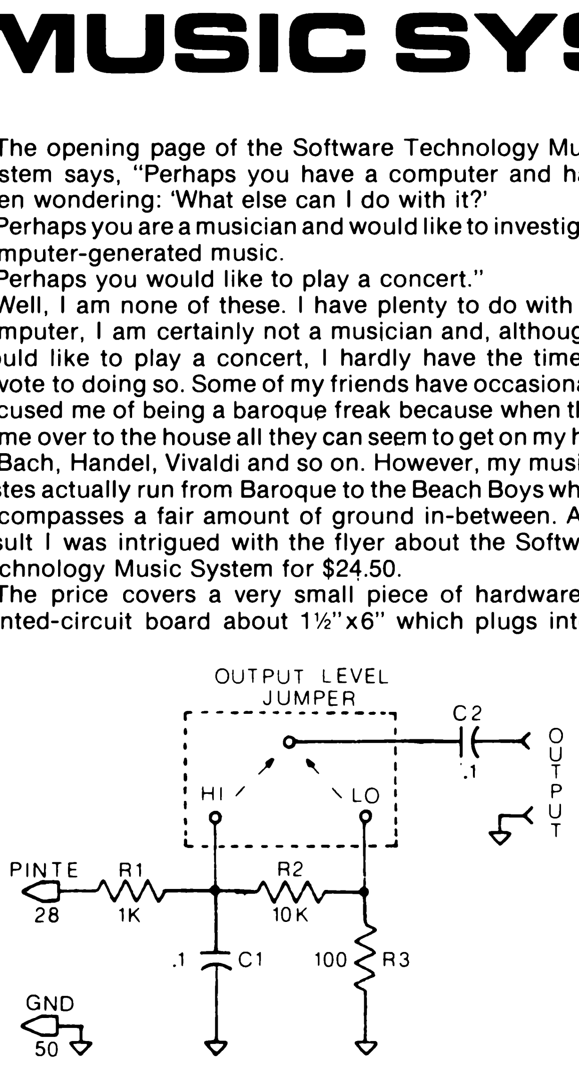


Fig. 1. The hardware is simplicity itself; the entire board has only five components.

My last investment in a music kit was also, curiously enough, in the same price range ($30). That was for a dulcimer which I purchased several years ago. The dulcimer might be termed all-hardware; perhaps hardwood would be a more accurate designation. In addition to the price I found several other interesting parallels between the dulcimer and the Software Technology Music System. The dulcimer has three strings — the music system has three voices. It took me, a totally inexperienced musician, a couple of hours before I could start picking out simple melodies on the dulcimer and making them sound decent using all three strings. Correspondingly, it also took a couple of hours before I felt relatively comfortable transposing music for the Software Technology Music System. The dulcimer has rather simple tuning, a two-octave range, an interesting tone if played in moderation, and is kind of a folksy musical instrument. I found the Software Technology Music System roughly the same. It's easy to tune, has a wider range (four octaves) with a nice tone although one certainly wouldn't want to overdo it, and it's kind of folksy — something that my wife and kids, who don't relate to computers all that well, found intriguing and could relate to. Enough of the parallels. What is the product like?

The printed-circuit board clearly is the model of simplicity. It goes together in about ten minutes — five minutes of which are spent letting the soldering pencil warm up. The addition of a shielded cable and a phono plug allows you to plug it into your hi-fi. In my case, since I was using a tv set for my display with my SOL-20 system, I decided it would be easy enough to tap into the volume control on the tv set and use its amplifier and speaker for the music system. Certainly not the fidelity of a good hi-fi system but adequate under the circumstances, plus the fact that it gives me a fully-transportable, self-contained system. In the way of other hardware, you need just the bare minimum system: an 8080-based CPU, with preferably 12K or 16K of memory since a typical musical tune takes approximately 1K of memory for each minute of the composition. If you don't want to overlay the compiled version of the tune on top of the source code you would certainly want at least 12K of memory for any tune more than a couple of minutes long.

The cassette tape of software comes in two different formats: a 1200-baud CUTS format on one side and on the other side the same programs in the 300-baud Kansas City format. The tape contains seven programs. First of all, the music program itself in object form, and six musical scores in source form including works such as the *Prelude in C Major* by J.S. Bach, *Bouree* by Handel and several other baroque pieces.

Upon assembling my system the first thing I did was check it out and make sure the tape read in correctly; in other words, I turned to page 24 in the manual, entered a few commands to my SOL-20 system to load and execute the music program, return to the SOLOS monitor and get the first demo program, then gave it the command "FILE" which verifies the new file just read in, then the command "SCORE" which compiles the file, and finally the command "PLAY" which plays it through the speaker system. Frankly, I was absolutely flabbergasted. The system loaded absolutely correctly the first time and played through my hi-fi system the first time. I had visions of spending the usual one or two evenings debugging the system, figuring out why it didn't load, why it didn't play, and so on, and frankly I was pleasantly surprised that the whole system loaded and played the first time.

Flushed with this extraordinary success I decided to try my hand at putting in a piece of music of my own. As I mentioned before, I am anything but a musician, nor are any of the people in my family musicians. Consequently the only music we had around were some things that were from the 1940's left over from when other people dropped over occasionally to play our piano and forgot to take their music home with them. In other words, our selection was rather limited. In this pile I found a piece of sheet music for *Yankee Doodle Dandy* by George M. Cohan in "an easy arrangement for piano." It suited my purposes perfectly because the arrangement was for three parts and the music board itself allows for three parts or three voices. I arranged things on my desk, the musical score in the center, a writing tablet and the manual for the music system on the left and started away. I first started by numbering the measures of the piece as was suggested in the music manual. As it turned out there were 42 measures in *Yankee Doodle Dandy.* Then I simply followed the directions and transcribed the music — the various notes, key signatures and so on from the sheet music to the appropriate computer notation. After completing eight measures I typed in the code and scored it, which is simply the command "SCORE" in the music system and played it. I expected to recognize the melody. Instead I found that I was playing some sort of funeral dirge. "Aha!" I said to myself. The speed must be wrong and so I tried to modify the speed by means of the command that fixes a certain number of computer cycles to a certain note length. However, affixing the minimum number of cycles to the longest note, in other words playing the piece as fast as I possibly could, while it didn't quite have the funeral quality when I started out, was still far too slow for comfort. My solution was to combine two measures into one and half the length of each of the notes. In other words, on the sheet music a quarter note became on my computer music an eighth note, a whole note became a half note and so on. This gave me the latitude that I needed and improved the tempo of the piece considerably. While *Yankee Doodle Dandy* is a rather simple piece, particularly compared to *Sarabande* which is the sample piece scored in the Music System manual, nevertheless, *Yankee Doodle Dandy* had several musical occurrances which were not in *Sarabande.* Consequently I found myself looking in the manual to find out how to do certain things. Clearly in a system such as this, once the hardware is functioning and the software is loaded in, it is the manual which becomes the central part of the system. Consequently a few comments on the manual are probably in order.

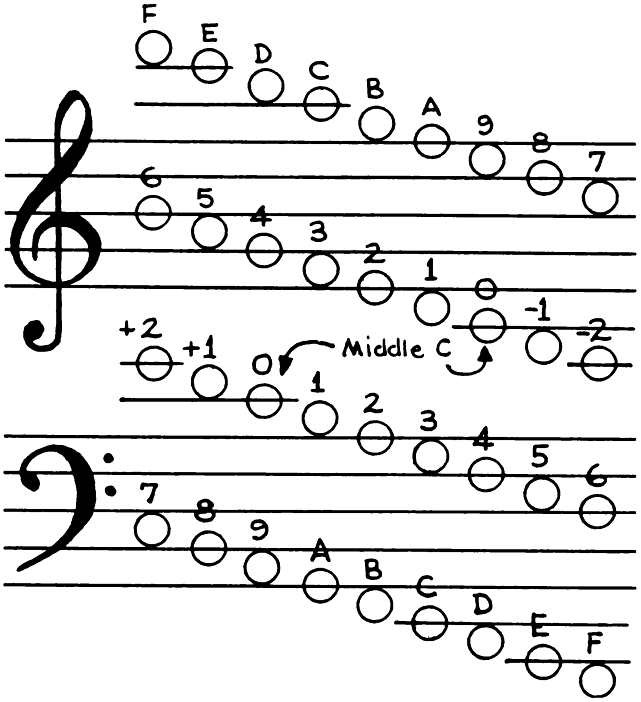


Fig. 2. Musical note tones are expressed in hexadecimal with notes on the treble clef being positive and on the bass clef negative.

It would certainly be helpful to have all of the symbols of the music system language defined in one place. As it is right now some of them are defined in the "commands" section of the manual, some others are defined in the "language summary" section, while still others are defined in the "musical note symbols" section, and lastly there are some symbols which are defined only in the text. The "rest" symbol, which is a dollar sign, turns out to be very important in *Yankee Doodle Dandy.* It's a rather minor part of *Sarabande* and is treated lightly in the text. On the other hand it would certainly have helped to be able to refer to some part of the manual and find out easily how to use the rest. Another thing which is not discussed in the manual is how to tie a note over from one measure to the next. The manual states that "since the computer always plays 'legato,' tied notes usually do not present much of a problem. However, a very soft, almost inaudible 'sh' sound is produced by the tone-generating routines whenever a new note is started." The discussion refers to two notes in the same measure. The question the user must ask himself, as I did, is "does that mean that when the same note is played one measure after another that it is also played 'legato' or does it mean that there is a bit of a pause or articulation between the two notes?" As it turns out, trial and error indicated that things are always played legato and if one wants a break between the same note played in two measures a very distinct articulation must be inserted at the end of the note played in the first measure.

After several hours wrestling with these various problems the system became clearer and clearer; trial and error was a most helpful friend for determining what would happen in different situations; and, as the manual says, "it will be a matter of blind luck or painstaking trial and error. Ultimately, the 'right' solution is the one that sounds best." The manual is absolutely correct in this regard. After several hours of trial and error I found that instead of transposing the music to a piece of paper and then into the computer that I could simply sit down at the keyboard and transpose the music directly from the musical score into the computer.

Later that day some friends were over to the house. This happened to be the Fourth of July and we had a few firecrackers to drink, not the exploding type (the drinkable type of firecracker is made from a jigger of whiskey in a tall glass with ice filled up with cranberry juice — a delightful summertime drink). In any event I turned on my music system, played some of the demonstration works and then said, "here is the first work that I composed myself" and, as those old ads say, they laughed when I sat down at my computer (and are probably still laughing yet!)

All in all, if you're willing to devote a couple of hours to putting together the hardware, playing the demo software, and learning how to transcribe some music with trial and error or whatever method you find most handy, you will find "The Music System" by Software Technology an extremely intriguing and worthwhile investment of $24.50.

For more information on the Software Technology Music System, write: Software Technology Corp., P.O. Box 5260, San Mateo, CA 94402. Phone 415—349-8080.

For more information on dulcimer (and other folk instruments) kits, write: Here, Inc., 410 Cedar Ave., Minneapolis, MN 55440. ■

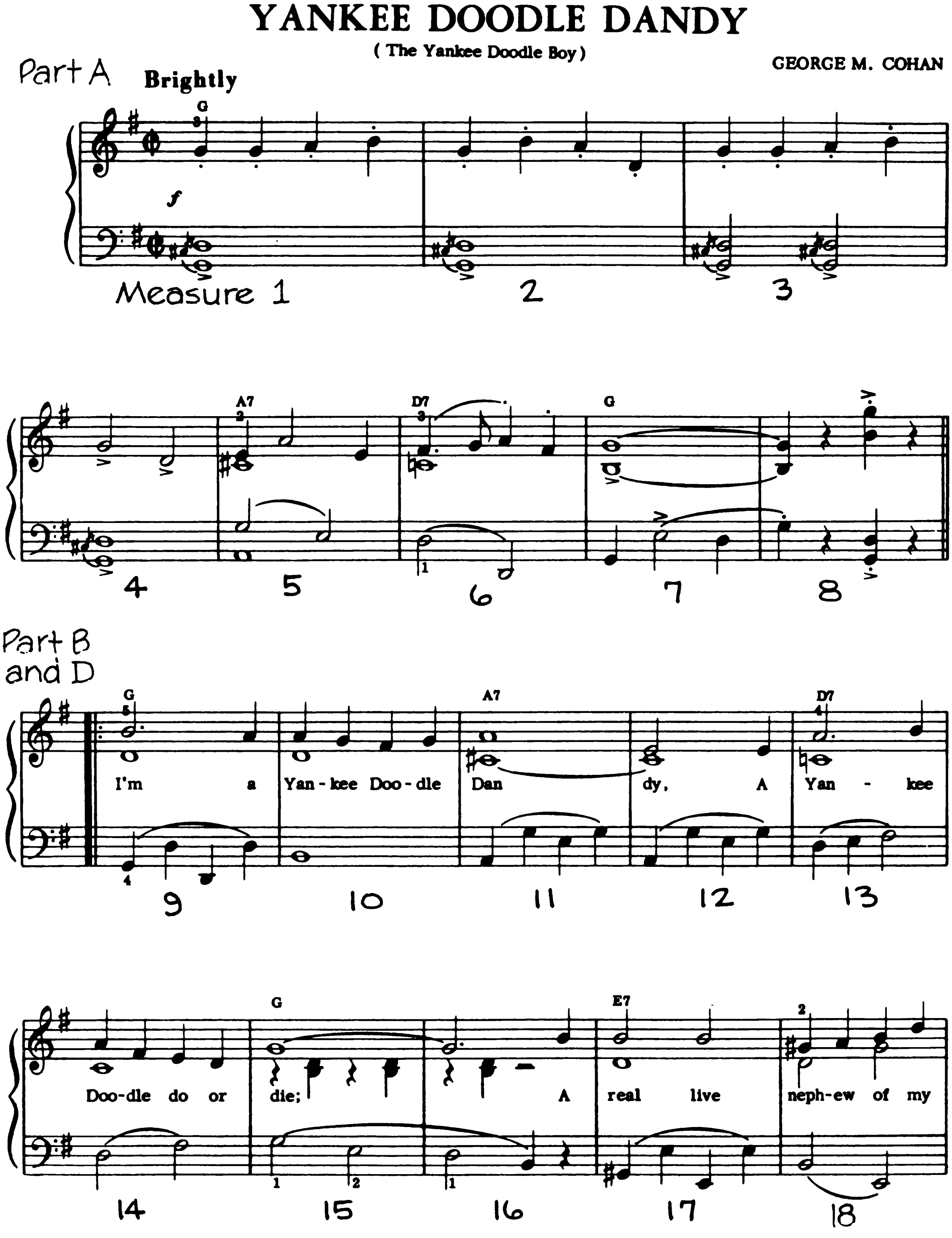
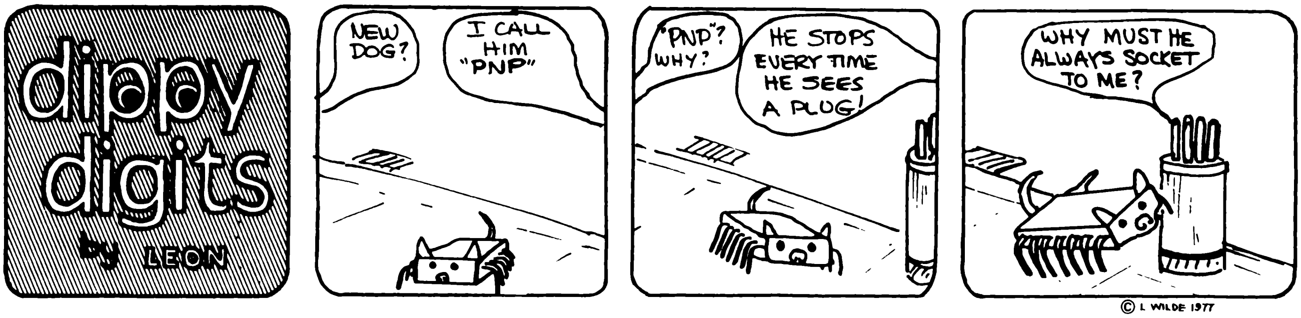

YANKEE DOODLE DANDY
(The Yankee Doodle Boy)
GEORGE M. COHAN
Part A
Brightly
G
A7
D7
G
Measure 1
2
3
4
5
6
7
8
Part B
and D
G
A7
D7
I'm a Yan-kee Doo-dle Dan dy, A Yan - kee
9
10
11
12
13
G
E7
Doo-dle do or die; A real live neph-ew of my
14
15
16
17
18

Fig. 3. First page of the piano score for *Yankee Doodle Dandy*.


dippy digits
by LEON
NEW DOG?
I CALL HIM "PNP"
"PNP"? WHY?
HE STOPS EVERY TIME HE SEES A PLUG!
WHY MUST HE ALWAYS SOCKET TO ME?
© L WILDE 1977

Music Transcription Example

A portion of the piano score of *Yankee Doodle Dandy* is shown in Figure 3. The computer source code is in Figure 4. Let's consider various lines of the computer code.

0020. A slash indicates a comment, and the rest of the line is ignored.

0040. 4 means that all the notes are transposed down four semitones. Thinking of the computer as a musical instrument the trick, as with any instrument, is to find the key which sounds best. In this case, four semitones down sounded 'right.'

0050. Defines the key signature, in this case one sharp.

0060. Defines the tempo. In this case, a sixteenth note (S) has 40 computer cycles. NS=90 would play the tune more slowly.

0080. Indicates that the measures that follow are one section of the piece (Part A). Part A will consist of everything until another P symbol. Note that Part D (line 310) is a repeat of Part B (RB).

0090. M1 defines the measure. It is ignored by the computer but is handy for debugging. A voice (or part) is a separate strand of music, in harmony or counterpoint, as in a three-voice fugue. Up to three voices may be defined in a measure, identified as V1 (assumed to be the first voice in a measure), V2, and V3. V1 usually carries the melody, or highest notes, V3 carries the bass, and V2 the mid-range voice.

Each note is represented by a pair of symbols, one of which defines the tone (position on the staff) and the other the length (whole, half, quarter, etc.). As mentioned in the main text, the note lengths were halved to give the piece adequate tempo, hence a quarter note in the music becomes an eighth note in the computer transcription. Also, one computer-music measure corresponds to two measures in the score. I4 means an eighth note of tone 4 (i.e., a treble G). The leading asterisk (*) indicates the treble clef. The comma following indicates that the note is to be played staccato. Voice 2 is in the bass clef ( ) and consists of two half notes of tone 6 (bass D). Voice 3 is also in the bass clef and consists of two half notes of tone A (bass low G).

Dotted notes are indicated by a period (.) after the letter indicating primary note length (see lines 0110 or 0140). A colon (:) indicates triplets. A dollar sign ($) indicates a rest (see lines 0120 or 0170).

Accidentals are indicated with a sign following the note to be modified, for example HO# (C sharp) and HO% (C natural) in line 0110.

0120. The length of a measure is equal to the longest voice in it. Shorter or undefined voices are filled with rests. Hence it is up to the user to keep track of beats and indicate rests for at least one voice to keep the proper tempo. Hence Voice 1 closes with I$ (eighth rest) in measure 8.

0140. Between measures 9 and 10 it is desirable to have an articulation (between "a" and "Yan" which are both A) otherwise they will sound almost as one long note. A double quote (") gives a long articulation equal to about two-thirds of a sixty-fourth note. The duration of the preceding note is reduced to compensate for the added rest.

Many other things that occur in various pieces of music can be handled in the Software Technology Music System. Even in this piece a better job could have been done with the short bass c-sharp in the opening four measures (instead of ignoring it). However, as with any instrument, proficiency comes with hours of practice.

```
0020 /YANKEE DOODLE DANDY BY GEORGE COHAN
0030 /COMPUTER TRANSCRIPTION BY DAVID AHL
0040 <4
0050 K1#
0060 NS=40
0080 PA
0090 M1 *I4,I4,I5,I6,I4,I6,I5,I1, V2@H6H6 V3@HAHA
0100 M3 *I4,I4,I5,I6,Q4Q1 V2@Q6Q6H6 V3@QAQAHA
0110 M5 *I2Q5I2I.3S4I5,I3, V2*HO#HO% V3@Q3Q5Q6QD
0120 M7 *H4I4I$IB,I$ V2*H-1I-1I$I6, V3@IAQ5I6I3,I$IA,
0130 PB
0140 M9 *Q.6I5"I5I4I3I4 V2*H1H1 V3@IAI6IDI6H8
0150 M11 *H5Q.2"I2 V2*HO#HO# V3@I9I3I5I3I9I3I5I3
0160 M13 *Q.5I6I5I3I2I1 V2*HOHO V3@I6I7Q8Q6Q4
0170 M15 *H4Q.4"I6 V2*I$I1I$I1I$I1 V3@Q3Q5Q6I8
0180 M17 *Q6"Q6I4#I5I6I8 V2*H1Q1Q4 V3@IA#I5ICI5Q8QC
0190 M19 *Q7Q6H5 V2*Q2Q1HO V3*I9I5IA#I5I9Q5"I5,
0200 M21 *I6Q5I4I2Q4I6 V2*VO# V3@Q3Q5Q3Q5
0210 M23 *H5Q.5I1 V2*HOQ.0 V3@I4Q6I5I4I5I4
0220 M25 *Q.6I5"I5I4I3I4 V2*H1H1 V3@IAI6IDI6H8
0230 M27 *H5H2 V2*VO# V3@I9I3I5I3I9I3I5I3
0240 M29 *Q.5I6I5I3I2I1 V2*HOHO V3@I6I5Q4Q6Q4
0250 M31 *V4 V2*I$I1I$I1I$I1I$I1 V3@Q3Q5Q6Q8
0260 M33 *I4,I4,I5,I6,I4,I6,I5,I1, V2@H6H6 V3@HAHA
0270 M35 *I4,I4,I5,I6,Q4Q1 V2@Q6Q6H6 V3@QAQAHA
0280 M37 *I2Q5I2I.3S4I5I3 V2*HO#HO% V3@Q3Q5Q6QD
0290 PC
0300 M39 *H4I4"I4I5I5# V2@H1I1 V3@IAI6I5I4I3
0310 PD RB
0320 PE
0330 M41 *H4I4I$IB V2*H-1I-1I$I6 V3@IAQ5I6I3I$IA
```

Fig. 4. Musical source code for *Yankee Doodle Dandy*.

Music Language Statements

Symbol	*Modifier*	*Meaning*
/	(None)	All characters on the rest of line are ignored.
P	Any letter (A-Z)	Define beginning of a part identified by the modifier. Any previous part is ended.
R	Any letter (A-Z)	Repeat the part named by the modifier.
M	Any character or characters	Define the beginning of a measure. Any previous measure is ended.
V	Digit (1,2, or 3)	All the notes following belong to the voice named by the modifier.
<	Hex Digit (0-F)	All the notes following are transposed down the number of semitones specified by modifier.
>	Hex Digit (0-F)	All the notes following are transposed up the number of semitones specified by the modifier.
*	(None)	Unless otherwise indicated, all notes following are assumed to be "+" (treble clef).
@	(None)	Unless otherwise indicated, all notes following are assumed to be "-" (bass clef).
↑	Signed Hex (+ or -) (0-F)	Transpose only those notes following that belong to the current voice up or down the number of whole steps indicated in the modifier.
K	Digit Char. (0-7) (# or &)	Key signature is defined by number and type (sharp or flat) specified in the modifier. If this symbol group is omitted, the key defaults to C major (no sharps or flats).
N	Char (H,Q,I,S)	Correlates the length of the note type in the modifier to the length of a beat.
=	2 Digit Hex (00-FF)	Equates the length of a beat to the number of internal cycles specified by the modifier.

Music System Commands

Commands can all be abbreviated to only the first letter. Portions in parentheses are optional operands.

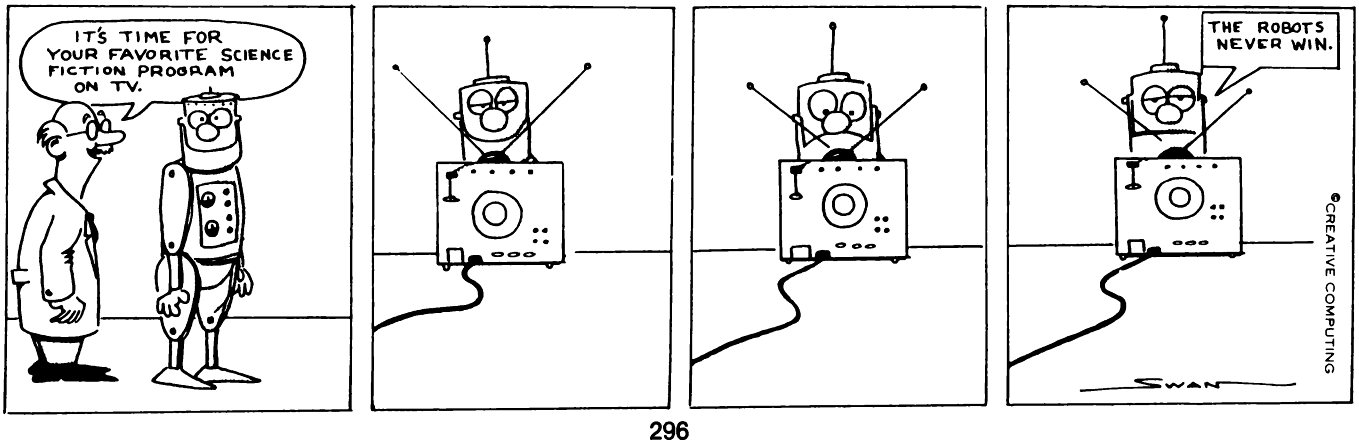
NEW (addr1)	Establishes a new file beginning at 'addr1'.
FILE (addr1)	Validates the file beginning at 'addr1'.
LIST (line 1 (line 2))	Displays the lines in the current file between 'line 1' and 'line 2'.
DELETE line 1 (line 2)	Deletes all lines between 'line 1' and 'line 2'.
SCORE (addr1)	The current file is compiled, and the resulting binary object code is entered into memory beginning at 'addr1'.
PLAY (addr1)	Generates musical tones according to the binary code at 'addr1',
RESERVE addr1	Marks the end of the memory space to be used by the music system.
RETURN	Returns to SOLOS (or to a properly written surrogate monitor).

Note Specifications

Note Modifier	*Name*	*Musical Example*
#	Accidental Sharp	#
&	Accidental Flat	♭
%	Accidental Natural	♮
'	Short articulation	none
"	Long articulation	none
,	Staccato	♩̣

Note Value Symbol	*Name of Note*	*Musical Equivalent*
W	Whole note	𝅝
H	Half note	𝅗𝅥
Q	Quarter note	♩
I	Eighth note	♪
S	Sixteenth note	𝅘𝅥𝅯
T	Thirty-second note	𝅘𝅥𝅰
X	Sixty-fourth note	𝅘𝅥𝅱

Note Value	*Name*	*Musical Example*	*Time Value Multiplier*
.	Dotted note	♩.	1-1/2
:	Triplet	♫	2/3
$	Rest		


IT'S TIME FOR YOUR FAVORITE SCIENCE FICTION PROGRAM ON TV.
THE ROBOTS NEVER WIN.
©CREATIVE COMPUTING
SWAN

A COMPARISON OF SOFTWARE SYSTEMS

In Volumes 1 and 2 of *Creative Computing* we ran an in-depth comparative review of every book on BASIC currently in print. While many sources review books, few, if any, compare them to each other. This group review was very well received.

Going one step beyond book reviews we've now embarked on a project to publish in-depth, comparative reviews of software and hardware/software systems. As with the books, these won't be based on manufacturer's literature or published specifications, but rather on our own exhaustive tests.

The first review of several BASIC language interpreters from this project appears in this issue. However, we are still in the formulation stage with the overall effort.

BASIC Language Comparisons

On what basis should a BASIC interpreter be compared? Number of statements? String functions? Matrix manipulation? Graphics commands? Level of FOR loop nesting? Richness of statements? File handling? Formatting? Timing functions? Probably all of the above and more. But with what emphasis? Frankly, that all depends upon your expected end use.

We've tried to come up with a number of reasonable comparative variables, although as this series of tests progresses, we'll surely think of more. If you have some ideas, please let Steve North or me know.

Hardware/Software Systems

Some time ago I got it into my head that it would be neat to devise a test routine that could be used on everything from the smallest microcomputer (or calculator for that matter) to the largest, giant number-cruncher. With that in mind I recalled "Different Digits," a problem we published back in Vol. 1, No. 2. Geoffrey Chase, D.O.M. at the Abbey, Portsmouth, RI, had devised a solution which he ran in five different versions of FOCAL, BASIC, FORTRAN and PAL. I wrote Geoff and asked him to devise a possible test based on this problem solution. He did so, and also sent a sheaf of expository notes on languages, EAE (hardware extended arithmetic element), weird features, and the like. Some of this appears in this issue, although most of it will be in the next issue.

I *think* we now have an algorithm/program (hopefully only the first of many) that can be run on a wide variety of machines in virtually any language that can really compare an 8080 MPU with PTCo BASIC to a PDP-8/E with Educomp 250 BASIC to a CDC 6600 with FORTRAN IV. But let us have your thoughts too. Remember, we're talking about potential execution speed differences of 10,000 to 1!! Remember too that an algorithm to compute, say SIN (X)/ SQR (X) 100,000 times is really not nearly as useful as an algorithm that uses a wide variety of statements, nesting, recursion, etc. and is long and complex enough to exercise the interpreter or compiler efficiency.

Since all computers are tens of thousands of orders of magnitude faster than a human, does it *really* matter how they compare to each other? And isn't price/performance ratio the crucial variable anyway? Stay tuned next issue and find out! —DHA

Review of Five Small Interpreters

Steve North

As someone around here keeps saying, "A computer without software might as well be a boat anchor" (—from Nautical Chinese Proverbs of Hing Yang Ahl). Whether or not you think computers have anything to do with boat anchors, the point is well taken—hardware is only half a computer system. It is possible to invest a good fraction of the total cost of a system in software alone. So what's the best software for you? Obviously you must take your own applications into consideration, but still, some interpreters are better buys than others. Since it isn't practical to try out every interpreter on the market, *Creative Computing* will start a series of reviews on languages for 8080-based microcomputers. If you want, we could also review 6800, 6502, Z-80, and F-8 software. Our reason for starting with 8080 software is that there are simply more products being offered and more claims being made in the 8080 market.

This issue we'll start with five small interpreters. Two of these can be had simply for the cost of reproduction, so it isn't really fair to compare them with products being offered for hundreds of dollars, but we'll do it anyway, to show that there is some very good software available for almost free. The five interpreters we're starting with are CASUAL, Palo Alto Tiny BASIC, Cromemco Control BASIC, Processor Technology 5K BASIC, and MITS 4K BASIC 4.0. Next time we'll get to some of the bigger full-feature BASICs.

Remember in using these reviews to take your own needs into consideration. Do you really care about computational speed if you just want to use your home computer for games? Probably not, since most simple games are I/O bound. Do you want a fancy text editor with your BASIC? Is it worth hundreds of dollars to you? do you want to use external data files in your application? Do you plan to try to control external devices with your computer? Or do you plan to use your computer solely for software development? We can collect facts, but in the end, it's your money....

A Review of the Review

Author, Price and Availability, and Size: Self-explanatory.

Reliability: Bugs we've found in our use of the software.

Documentation: There are two types to look for—operating instructions, and source listings of the inner works of the language. The first type is essential; the second is of use to you only if you understand machine code and wish to hack around with the language, modify it, or just understand how it works.

Speed: We used a test program that finds two whole numbers such that the two numbers and their product contain all the decimal digits (0-9) with no repetitions. For more details on the test program see Geoffrey Chase's article right after this, and page 66 of the May-June 1976 issue of *Creative.* Since interpreters that use 15-bit numbers (and a sign bit) for their math overflow when the step for checking 34 is reached, we'll run the test to 34, and up to 55 on the interpreters with floating-point math. If readers send in their own test programs we will consider using them. In the next installment of this review, we'll use another test program to check the speed and accuracy of transcendental functions.

Features: We'll list and sometimes describe the features of each interpreter. Of course we can't explain everything (the MITS BASIC manual alone is longer than a single issue of *Creative*), so make the following assumptions for the sake of simplicity:

(1) All the interpreters have a line editor (which permits one to enter, change, or delete lines by line number), character backspace, and line-delete functions.

(2) All the interpreters have a "Direct" mode of execution. If you type a statement without a line number, it is done immediately. If you typed

PRINT A,B,C

the computer would immediately respond by typing the values of A,B and C. This can be a handy debugging aid.

(3) All the interpreters permit you to break a program by typing a character (usually control-C) at the keyboard.

(4) Multiple statements are permitted on the same line. (Example: 100 PRINT A: LET Q=23: GOTO 370)

(5) A full set of six relational operations.

And finally, our User Comment.

CASUAL

Author: Bob Van Valzah

Size: 1.6K

Price and Availability: A paper tape can be obtained from the CACHE Software Library, Lloyd Smith, 530 Pierce Avenue, Dyer, IN 46311

Reliability: No bugs.

Documentation: An almost complete copy of the instructions and a source listing of CASUAL can be found in *Dr. Dobb's Journal,* Vol. 1, No. 10. The remainder is in *DDJ* Vol. 2 No. 2.*

Speed: 40 minutes to Step 34. (Stopped by operator—CASUAL does not check for overflow in arithmetic operations.)

Features: CASUAL is not BASIC, although it is BASIC-like. Instead of using keywords such as PRINT or GOTO, CASUAL uses symbols such as '.', '$', and '?'. There is, however, a close correspondence between what CASUAL can do and what BASIC can do.

Commands: NEW, LIST, RUN, O (jumps to Operating System,) TAPE and SAVE (for loading programs from mass-storage devices.)

Statements: There are only three types of statements in CASUAL:

(1) ?expression (same as BASIC PRINT expression)

(2) leftside = rightside (same function as BASIC LET, INPUT, GOTO, GOSUB, RETURN, IF/THEN, DIM, and almost everything else in BASIC.)

(3) expression (string input statement)

You may at this point be wondering how CASUAL uses an equivalency for all those things. GOTO and IF/THEN, instance, are implemented with a '.=' structure. '.' is always the value of the next line to be executed..=500 is the same as a BASIC GOTO 500..= -1 will cause CASUAL to return to the line editor (similar to BASIC STOP or END). .=0 will fall through to the next statement. To write IF X=23 THEN 400, you'd code .=400*(X=23), which will evaluate to either 0 or 400. There are other symbols in CASUAL for calling subroutines, peeking and poking memory, direct I/O to any port, and other interesting things.

Variables: A-Z, 15-bit signed integers (0 - ± 32767)

Functions: +, -, *, /

User-defined functions: The '↑' symbol is the user-defined function. It can be defined more than once in a program. It has no argument, so any variables used in it are common to the whole program.

Arrays: There are two arrays in CASUAL: a single-byte array (for value between 0 and 255) and a double-byte array (for regular CASUAL numbers.) The arrays are one-dimensional and are referenced with "(subscript) for the double-byte array, and '(expression) for the single-byte array. Before using the arrays you must define their starting addresses in abolute memory.

Machine-language subroutine interfacing: The machine-language subroutine whose address is found at location 3 is called when the '@' symbol is found on the righthand side of an expression. Parameters can be passed both ways.

Character strings: In a rudimentary way. When inputting a string, it is placed character by character in the single-byte array. It can then be processed numerically, or output using '>', CASUAL's equivalent of the BASIC CHR function.

Formatted print: None.

Editing function: None.

External files: None.

Error messages: Seven, indicated by number.

Extra stuff: Version .186 has TAPE and SAVE drivers for Tarbell cassettes. A CASUAL-VDM DRIVER is in the works. There are many features of CASUAL we haven't mentioned here—the best way to find out about them is to get a copy. CASUAL has an initializaton routine similar to MITS' which permits you to delete some CASUAL features in return for more memory. CASUAL has no subroutine nesting and does not have < = or > = relational operators.

User Comment: It's nice (finally!) to see some non-BASIC languages for microcomputers. While CASUAL is a minimum language in some respects, and it may take some

---

In case you're a hermit, *Dr. Dobb's Journal of Computer Calisthenics and Orthodontia* is an excellent source of free software and programming hints. It concentrates mostly on low-level stuff. One issue is $1.50; one year (ten issues) is $12. *Dr. Dobb's Journal,* Box E, Menlo Park, CA 94025.

practice to use it effectively, it certainly does squeeze a lot of performance out of 1.6K of memory! Anyway, CASUAL is fun to use.

Palo Alto Tiny BASIC

Author: Dr. Li Chen Wang

Price and Availability: Paper tape can be obtained for $4, from Community Computer Center, 1919 Menalto Avenue, Menlo Park, CA 94025

Size: 2.3K

Reliability: No problems.

Documentation: A complete set of instructions and source listing appeared in *Dr. Dobb's Journal,* Vol. 1, No. 5. Vol. 2, No. 2 contained extensions for simple character strings, calling machine-language subroutines, and PEEK and POKE-ing memory.

Speed: 31 minutes to Step 34.

Features:

Commands: RUN, LIST, NEW

Statements: FOR/NEXT, GOSUB, GOTO, IF/THEN (followed by any statement), INPUT (with optional prompt string), LET, REM, RETURN, STOP

Variables: A-Z, 15-bit signed integers (0 to ± 32767)

Functions: +. -, *, /, ABS, RND (returns a value between 1 and the argument), and SIZE (returns the number of free bytes in the system).

User-defined functions: None.

Arrays: PATB has one array, named '@'. It has a single subscript and uses all available memory (so it is not dimensioned).

Machine-language subroutine interfacing: None in the original PATB.

Character strings: None in the original PATB.

Formatted print: Printing '#integer' sets the number of spaces in which an expression is printed.

Editing features: None.

External files: None.

Error messages: Three: WHAT? in response to a syntax error; HOW? in response to overflow, a missing line number, etc.; and SORRY when there is insufficient memory.

Extra stuff: Typing control-O stops all output until another control-O is entered, which is handy for entering paper tapes. PATB statements can be abbreviated: G. for GOTO, GOS. for GOSUB, E. for FOR, and so on.

User Comment: Palo Alto Tiny BASIC is an elegantly simple Tiny BASIC. It is the kind of Tiny BASIC you would expect to find on a big computer (if big computers had Tiny BASIC)—there's nothing you can do to bomb the system! PATB is powerful enough to run Star Trek in just 8K of memory (including PATB itself)! Far out! The price seems reasonable enough.

Cromemco Control BASIC

Price and Availability: $15 for a paper tape. from Cromemco, 2432 Charleston Rd., Mountain View, CA 94043.

Size: 3K, located at E400 hex.

Reliability: No problems.

Documentation. Very complete instructions are provided, but no source code.

Speed: 40 minutes to Step 34

Features: Same as Palo Alto Tiny BASIC, with the following additions:

Commands: LOCK (allocates area in memory for program files), SAVE (saves files in locked area), EPROM (burns in a file on an EPROM with a Cromemco Bytesaver), LOAD (loads file—opposite of SAVE), WIDTH (sets terminal width), NULL (number of nulls after a CR/LF), QUIT (jumps to O/S).

Statements: PUT (stores a value in absolute memory), OUT (outputs to an I/O port).

Variables: AO-ZO. Hex literals are denoted by a leading % sign, as in %F5AB.

Functions: +, -, *, /, SGN, AND, OR, XOR, GET (gets a value from absolute memory), IN (input from an I/O port), LOC (to find absolute address of array elements), CALL.

User-defined functions: None.

Arrays: Has one other array in addition to @(X), named &(X), single-byte array. It overlaps with @(X) in memory.

Machine-language subroutine interfacing: Through the CALL functions. Parameters are passed both ways through the stack.

Character strings: Strings of up to 132 characters can be accessed by $(X), the way numbers are accessed by @(X) and &(X). String and array space intentionally overlap. There is a string input statement.

Formatted statement print: Same as PATB.

Editing functions: None.

External files: No data files. As mentioned before, program files can be kept in high memory with LOCK, LOAD, SAVE, and EPROM commands. These programs can be called by other programs with a special version of the RUN command. A STOP in a called program will function as a return to the calling program.

Error messages: Same as PATB.

Extra stuff: None, other than the extensions already mentioned.

User comment: Cromemco Control BASIC is an expanded version of PATB, primarily intended for process-control applications. For this reason it seems to be more hardware-oriented, while the original PATAB does not get the user involved in knowing absolute memory locations, hex math, or I/O ports. If you need to control devices, fine, get CCB; if not, stick with PATB, which is smaller and faster. It is unfortunate that CCB is available only in copies that run at E400 (not zero), which many people will find to be a major obstacle in their use of CCB, unless they want to burn it into a few PROMs on a Bytesaver.

Processor Technology 5K BASIC

Authors: Processor Technology and Applied Computer Technology.

Price and Availability: Available on CUTS cassette for $14.50, from Processor Technology Corp., 620 Hollis St., Emeryville, CA 94608.

Size: 6.6k

Documentation: The 40-page instruction manual explains the features of the language with frequent examples. The source code is not available.

Reliability: There are no known bugs. (An earlier release of BASIC/5, called 5K BASIC, did have some problems, but these have been cleared up).

Speed: 26 minutes to Step 34; 65 minutes to Step 55.

Features:

Commands: CLEAR (clears variables), GET (loads program from cassette), EDIT (enters edit mode), LIST, NEW, RUN, XEQX (loads and executes a program).

Statements: DIM, RESTORE, READ, DATA, LET, FOR/NEXT, IF/THEN, STOP, END, GOTO, INPUT, PRINT, GO-SUB, RETURN, FILE (opens a data file), CLOSE (closes a data file), SET (to set input port, output port, speed of video display, memory size, and nul characters after a CR/LF.

Variables: A-Z,A0-Z9, floating-point numbers (.1E-127 to .999999E-127).

Functions: +, -, *, /, ABS, INT, SGN, RND, SQR, SIN, COS, TAN, TAB, ARG, CALL.

User-defined functions: None.
Arrays: Yes, one-dimensional.
Machine-language subroutine interfacing: With ARG and CALL. ARG is used to set up parameters to be passed to the machine-language program. The CALL returns a value from two of the CPU registers.
Character Strings: None.
Formatted Print: Yes, by use of %specifications% in a PRINT. This can be used to set the number of trailing zeros, number of places to the right of the decimal, exponential format, or freeform.
Editing Functions: An edit mode permits insertion and deletions of characters on a program line. Control characters (or special keys on a SOL-20) are used to control editing functions. Simple, but powerful.
External Files: BASIC/5 has provision for reading and writing data files using a Processor Tech CUTS cassette interface. The syntax required to use this feature is similar to disk file I/O in some BASICs, making it rather powerful.
Error messages: Thirteen, by two-letter error codes.
Extra stuff: 5K BASIC has a built-in VDM driver, with some interesting features: it can output at a variable rate of speed, halt entirely, turn the cursor on or off, or clear the screen.

User Comment: Processor Tech BASIC/5 is a cheap, powerful BASIC. We especially like the cassette-file I/O, which is a lot better than reading or writing data byte-by-byte. The only thing we miss is multidimensional arrays, but in view of the low price and nifty features of BASIC/5, we'll overlook that. If you don't have a CUTS cassette interface you may find BASIC/5 to be of more limited usefulness since you won't be able to use the cassette-file I/O.

MITS 4K BASIC 4.0

Author: Microsoft.
Price and availability: $150 (only $60 for owners of Altair computer, 4K RAM, and I/O module). Paper tape or MITS cassette.
Size: Something over 3K.
Reliability: No problems.
Documentation: Extensive operating instructions are provided. Unfortunately, MITS lumped the documentation for all their BASICs together, so you must scan past all the documentation on 8K BASIC and Extended BASIC. No source code is available.
Speed: 29 minutes to Step 34; 74 minutes to Step 55.
Features:
Commands: CLEAR (clears variables), LIST, NEW, RUN.
Statements: DATA, DIM, END, FOR/NEXT, GOTO, GOSUB, INPUT, LET, IF/THEN, PRINT, READ, REM, RESTORE, RETURN, STOP.
Variables: A-Z, A0-A9, floating-point numbers (1.70141E+38 to -2.9387E-38).
Functions: +, -, *, /, ABS, INT, RND, SGN, SIN, SQR, TAB, TAN, USR.
User-defined functions: No.
Arrays: Yes, one-dimensional.
Machine-language subroutine interfacing: Through the USR function. Parameters can be passed both ways.
Character strings: None.
Formatted print: None.
Editing functions: None.
External files: None.
Error messages: Twelve, by two-letter error codes.
Extra stuff: Has an initialization dialogue permitting deletion of unwanted features to save space. BASIC patches itself for MITS interface boards depending on the initial position of the sense switches. We had some difficulty using 4K BASIC with a non-MITS interface and eventually just patched out the routine which fixed up the I/O in BASIC. 4K BASIC will not run on a Z-80 CPU.

User Comment: A small, solid floating-point BASIC. The price seems rather prohibitive for people who don't own Altairs, however. If you own an Altair and the right options, and don't plan to do anything too fancy with your computer, you might consider getting 4K BASIC.

Some Notes on Languages and Another Test Installation

Geoffrey Chase

HARDWARE

16K PDP-8/E with RK05/RK8E cartridge disk, TD8/E Dectape, high-speed reader/punch for paper tape. Console terminal (there are others): LA-36 Decwriter.

The machine has the EAE hardware boards that enable a 1-by-2 multiply and a 1-into-2 divide, both integer operations, as well as certain other operations such as normalize.

The machine does not have DEC's floating-point processor hardware (FPP).

The benchmarks were taken under stand-alone OS-8.

Fortran-2

The Fortran-2 (DEC) takes no account of EAE hardware. All operations are software—simulated FPP, if you will. The floating format carries 27 bits of precision.

The "hybrid" program uses machine-language patches to speed two things only (more could be done): integer division, and subscribing.

The resulting code is nearly twice as fast. It is also shorter—the object code, that is!— even though the source code (what the programmer wrote) seems to be longer. One Fortran statement can generate many lines of object (binary) code.

The Fortran-2 system uses as a second pass the SABR relocatable assembler. Lines beginning with "S" are passed, uncompiled, to the assembler directly.

The resulting code is less than optimum. There are many, often unneeded, changes or resets of memory field (core stack); these are especially baneful under ETOS or similar systems which must interpret the virtual (user's specified) memory field into the physical (user's core partition) memory field.

There are very many "external" subroutine calls, even if the user never says CALL or defines a FUNCTION. In particular, the overhead of the subscripting routine is high.

In almost any language, *some* price is paid for the luxury of subscripts.

The Floating Fix Problem

PDP-8 Fortran-2 is limited to signed integers between -2048 and +2047. Even larger systems not infrequently allow values up to say, around 16,000; not high enough for the AHLDIG program.

Small floating values can be truncated in any Fortran by

INT = VALUE or INT = IFIX(VALUE)
VALUE = INT or VALUE = FLOAT(INT).

But floating values too large to be true integers pose a nasty problem. One solution (machine-dependent) is shown in the Fortran-2 AHLDIG.F2: create a suitable *de*-normalized 0.0 and add it to VALUE.

At the Fortran-2 programming level, without recourse to assembler language, one could try the following:

(1) find out how many binary bits of precision are carried in your floating-point ("REAL") numbers. These are often termed, rather inaccurately, "mantissa" bits. The sign bit (+ or -) doesn't count;

(2) early in the program and before repeated loops, define M = number of "mantissa" bits, and define DENORM = 2.0** (M-1).

Then, where the present Fortran-2 source code does a QUOT = QUOT + ZERO, write instead QUOT = (QUOT + DENORM) - DENORM. The parentheses are usually necessary.

This is slower than the highly machine-dependent "ZERO" code shown, but it should do the job.

Fortran is pretty rigid. There is no equivalent to BASIC's INT ( ), nor is recursion allowed in subroutines. About its only virtue—at the Fortran-2 level—is speed, and the ability to insert binary code.

Note: Strictly "classic" Fortran requires:

(1) all line numbers to be indented one space;

(2) all WRITE formats to start with a carriage-control character, for example 1H (space), before all else in the FORMAT.

(3) device numbers ("WRITE(1,200)...") differ wildly from one system to another.

Fortran-4

The DEC Fortran is much more sophisticated compiler than the Fortran-2. It does *not* allow—typical of Fortran-4's—inserted binary code. The output is highly optimized code suitable for assembly by the RALF assembler. Integer (fixed) values may range up to roughly 8,000,000.

The fly in the ointment: RALF assembler code for execution by FPP *hardware*. If, like this writer, you haven't this rather costly option, then the run-time system must *interpret* the RALF code. So we come full circle, from compilation to interpretation: from speed to slowness.

The run-time system does sense the presence or absence of the EAE multiply/divide, etc., hardware. My timings on Fortran-4 are thus very much slower than would be obtained on an FPP machine, but faster than would be obtained on a machine without EAE hardware.

Educomp OS-8 BASIC

This is a compiled language with an interpretive run-time system that makes some use of EAE hardware if present. Not surprisingly, the run-time is not too much slower than Fortran-4's (less than twice the run time). If, like PDP-11 RSTS, a true integer mode were available, the two languages might wind up in a near-tie.

Edu-20 BASIC (DEC)

This is maybe a "worst case" for purely interpretive languages. FOCAL and the Educomp EDU-200/500 would almost certainly be faster.

Still, pure interpreters are instrinsically slow when number-crunching is required. All will show a very marked difference in time relative to any of the preceding languages.

(It might be mentioned that FOCAL—not shown here—has been considerably expanded and to some extent speeded by non-DEC programmers. J. van Zee at the Chemistry Dept., University of Washington, Seattle, currently markets a "U/WFOCAL" that is a large superset of DEC FOCAL and runs faster.)

Summary of Timings

	TO '55'		TO END		
	Hr	*Min*	*Hr*	*Min*	
Hybrid		2.2		4.9	
Fortran-2		4.1		9.4	
Fortran-4		10.1		23.9	
Educomp Basic		17.3		40.8	
Edu-20 Basic	5	44.8	13	35.0	est.

Hybrid by author; Edu-Basic from Educomp (Hartford); Others from DEC (Maynard)

Program Listing: Fortran-4

```
C    'AHLDIG.F4'     PAS      3/77

     INTEGER DIGIT(10),CHAR(3),A,B,C,A1,A2,QUOT

     DATA CHAR /'A','B','C'/

     DO 490 A = 12,98
     WRITE (4,100)A
     A1 = A/10
     A2 = A - 10*A1
     IF (A2 .LT. 2) GO TO 490
     IF (A2 .EQ. A1) GO TO 490
     MODA = MOD(A,2)

C    ...WHICH GIVES MODA = 1 IF A2 IS ODD, ELSE 0

     ITEMP = 10000/A

     DO 470 B = ITEMP,987

     DO 230 I = 1,10
230  DIGIT(I) = 0

     DIGIT(1+A1) = 1
     DIGIT(1+A2) = 1
     QUOT = B/10
     J = B - 10*QUOT
     IF (J .LT. 2) GO TO 470
310  MODB = MOD(B,2)
     IF (A2+MODB .EQ. 6) GO TO 470
     IF (J +MODA .EQ. 6) GO TO 470

C    ...KNOCKS OUT 5*ODD AND 6*EVEN

     IF (DIGIT(1+J) .GT. 0) GO TO 470
     DIGIT(1+J) = 2

     ITEMP = QUOT
     QUOT = QUOT/10
     J = ITEMP - 10*QUOT
     IF (DIGIT(1+J) .GT. 0) GO TO 470
     DIGIT(1+J) = 2
     IF (DIGIT(1+QUOT) .GT. 0) GO TO 470
     DIGIT(1+QUOT) = 2
```

```
0040          C = A*B
0041          ITEMP = C

0042          DO 400 I = 1,4
0043          QUOT = ITEMP/10
0044          J = ITEMP - 10*QUOT
0045          IF (DIGIT(1+J) .GT. 0) GO TO 470
0046          DIGIT(1+J) = 3
0047          ITEMP = QUOT
0050    400   CONTINUE
0051          IF (DIGIT(1+QUOT) .GT. 0) GO TO 470
0052          DIGIT(1+QUOT) = 3

        C     SUCCESS:

0053          WRITE (4,150)A,B,C

0054          WRITE (4,110) (CHAR(DIGIT(I)),I = 1,10)
0055          WRITE (4,100)

0056    470   CONTINUE

0057    490   CONTINUE

0060          STOP

0061    100   FORMAT(1H ,1I3)
0062    110   FORMAT(1H ,10A1)
0063    150   FORMAT(1H0,2I7,F8.0)

0064          END
```

Program Listing: Educomp OS-8 BASIC

```
100 REM.       "AHLDIG.BA" V.4 PAS      3/77
110 DIM D(9)
120 !  ----------
130 FOR A=12 TO 98
140 PRINT A
150 A1=INT(A/10)
160 A2=A-10*A1
170 IF A2< 2 THEN 510   ! ENDS IN 0 OR 1
180 IF A2=A1 THEN 510
190 M1=A2-2*INT(A2/2)   ! 1 IF ODD, 0 IF EVEN
200 !  ======
210 FOR B=-INT(-10000/A) TO 987
220 FOR I=0 TO 9
230 D(I)=0
240 NEXT I
250 D(A1)=1
260 D(A2)=1
270 V=B
280 F=2
290 Q=INT(V/10)
300 J=V-10*Q             ! REMAINDER=LOWEST DIGIT
310 IF J<2 THEN 490      ! 0 OR 1
320 M2=J-2*INT(J/2)      ! 1 IF ODD, 0 IF EVEN
330 IF A2+M2=6 THEN 490          ! 5*ODD OR ELSE 6*EVEN
340 IF J+M1=6 THEN 490           ! SAME CHECK
350 GOSUB 570
360 IF F<0 THEN 490
370 C=B*A
380 F=3
390 V=C
400 GOSUB 550
410 IF F<0 THEN 490
420 PRINT " A"," B"," C"
430 PRINT A,B,C
440 FOR I=0 TO 9
450 PRINT CHR$(64+D(I));
460 NEXT I
470 PRINT
480 PRINT
490 NEXT B
500 !  ======
510 NEXT A
520 STOP
530 !  ----------
540 V=Q
550 Q=INT(V/10)
560 J=V-10*Q
570 IF D(J)=0 THEN 600
580 F=-1
590 RETURN
600 D(J)=F
610 IF Q>9 THEN 540
620 IF D(Q)>0 THEN 580 ! AS BEFORE
630 D(Q)=F
640 RETURN
650 !  ----------
660 END
```

Timing Run: Educomp OS-8 BASIC

```
EXECUTE AHLDV4
 12
 13
 14
 15
 16
 17
 18
 19
 20
 21
 22
 23
 24
 25
 26
 27
 A             B              C
 27            594            16038
CCACBBCACB

 28
 29
 30
 31
 32
 33
 34
 35
 36
 A             B              C
 36            495            17820
CCCABBACCB

 37
 38
 39
 A             B              C
 39            402            15678
BCBABCCCCA
```

```
 40
 41
 42
 43
 44
 45
 A             B              C
 45            396            17820
CCCBAABCCB

 46
 A             B              C
 46            715            32890
CBCCABABCC

 47
 48
 49
 50
 51
 52
 A             B              C
 52            367            19084
CCABCABBCC

 53
 54
 A             B              C
 54            297            16038
CCBCAACBCB

 55
 56
 57
 58
 59
 60
 61
 62
 63
 A             B              C
 63            927            58401
CCBACCABCB
```

```
 64
 65
 66
 67
 68
 69
 70
 71
 72
 73
 74
 75
 76
 77
 78
 A             B              C
 78            345            26910
CCCBBBCAAC

 79
 80
 81
 82
 83
 84
 85
 86
 87
 88
 89
 90
 91
 92
 93
 94
 95
 96
 97
 98

STOP AT LINE 520
READY

40 MINUTES, 47 SECONDS
```

A DYNAMIC DEBUGGING SYSTEM

Steve North

Are your eyes weary from staring at blinking lights as you attempt to debug a program the hard way? Or are you fed up with the tedium of using a simple debugging monitor with only breakpoints to help you? If you own an 8080-based microcomputer, then relief is in sight. Relief, in this case, is the Dynamic Debugger, written by David Benevy and marketed by the Computer Mart of New Jersey.

The Dynamic Debugging System (DDS) is a software debugging facility for 8080 assembly-language programs. DDS is actually a program which helps you to debug your own program. A simple debugging monitor lets you enter breakpoints in your program, runs your program for awhile, and then lets you see the condition of the registers and program when the breakpoint was hit. DDS, on the other hand, monitors the execution of your program as it is going on, displaying pertinent information and identifying possible problem areas. It uses a softcopy device for the display (versions are available for the Processor Technology VDM-1 and popular CRTs, but the VDM version is better, since information may be updated almost instantaneously. DDS can only be used on CRTs with cursor addressing.) A keyboard is used to communicate with DDS.

The Dynamic Debugger has three types of functions. The first type controls the display of the program being debugged. Actually, most of the display is handled automatically. At the middle of the top of the screen, DDS displays the current program counter and the next five instructions to be executed, in easy-to-read mnemonic format. To the right of this is a display of all the CPU registers, the words they point to, both as hex and ASCII characters. The individual bits in the flag word are also broken up into mnemonics. In the 64-column version they are displayed as single characters (C for carry, for instance). In the 80-column version, the entire name of the flag (such as CARRY) is displayed if the flag is on. To the left side of the top of the screen, DDS displays the current stack pointer, the top five words on the stack, and the words they point to, as hex and ASCII characters. DSS also maintains its own stack of valid return addresses, which is displayed to the far left of the screen. So at a glance you can see exactly what's going on in all the registers and the stack, and have a pretty good idea what your program is doing too.

Now for the part of the display which the user controls. There are two types of displays you may ask for. One of these is a hex/ASCII display of selected portions of memory. There are six lines of display available in the 16-line version, and 11 lines in the 24-line version. These lines may be split up as you wish. For instance, lines 1 and 2 could display memory from 3000 to 301F, while lines 2 through 6 display memory from 2000 to 203F. You may also display memory as instructions instead of raw data. Two such groups of instructions may be displayed simultaneously. The instructions are displayed in mnemonic format similar to the display of the code currently being executed at the top of the screen. When selected, these displays are located in the lower half of the screen. These displays are requested by DM (display memory) and DI (display instruction) commands. The two bottom lines on the screen are used for displaying error messages and echoing keyboard input, respectively.

The second group of commands is used for manipulating the user's program or other data. Using the keyboard, you may enter bytes, words (groups of two bytes), or characters into memory; fill portions of memory with a constant; move blocks of memory, or find a combination of up to three bytes. You may also modify any of the registers or register pairs including the program counter and stack pointer, the stack itself, or the Dynamic Debugger's separate stack of return addresses. So you also have complete control over everything going on in your computer.

The third and most powerful set of commands is used to control and execute the program to be debugged. When debugging a program with DDS, your program never actually takes control of the system. Instead, DDS executes your program step by step, updating the display as often as you wish. (This is why it's nice to use a VDM-1. The VDM can be updated very quickly but CRTs may take a while longer.) As your program runs, you can see the program counter advance, registers change, data get pushed on and popped off the stack, or portions of memory change.

Stepping

One character on the keyboard (ESCAPE) is used as the stepping character. Every time this key is entered, part of the user's program is executed by the DDS. The exact number of instructions executed depends on the value you enter with the STEP command. By entering a step of one, the program may be single-stepped by hitting the stepping character. Or you may enter a larger step and run longer sections of code.

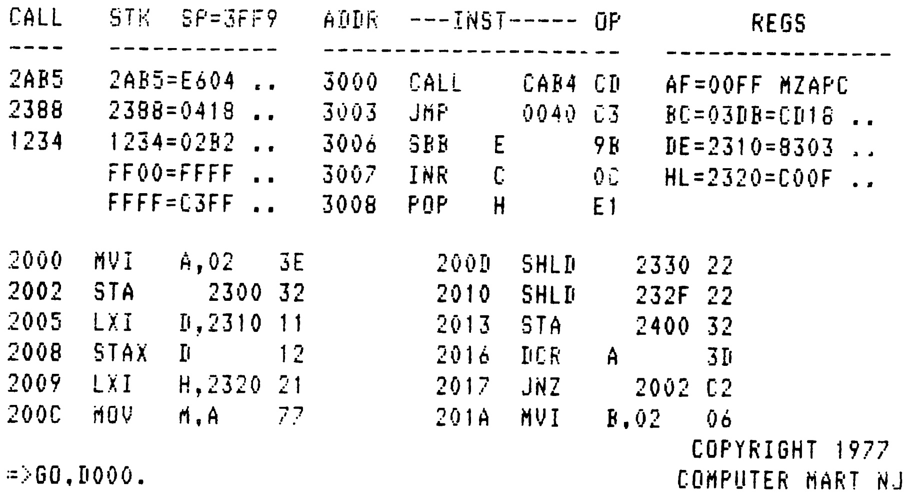
You may also set the time delay between instructions, which permits you to execute the program slowly while referring to a source listing. Execution of your program stops when DDS executes the specified number of instructions or when you enter a key (it doesn't matter which one).

Detection

The Dynamic Debugger gets even better: it not only tells you what's going on, it stops when it detects something wrong. For instance, if you PUSHed more data on the stack than you POPped in a subroutine, there would normally be an invalid return address on the stack and your program could run off into the woods. By maintaining its own list of valid return addresses, DDS will stop when it thinks you're trying to use a bad return address. Since some programmers do tricky things with the stack such as intentionally pushing extra words on the stack to set up return addresses, this feature may be disabled. DDS also halts if you attempt to execute an undefined opcode, a HLT, or a RST.

Error Condition

DDS also permits you to enter your own error conditions. For instance, you can tell DDS to stop executing your program if the program references data outside of a certain range, if the program reaches a specified address, if certain opcodes are executed, if the stack goes outside of a given range, if the program attempts to store in a certain location, etc. Note that for most of these, several conditions may be active at once. For instance, up to five opcodes may be used in the opcode stop option. When DDS finds an error, it displays an error message on the error line and stops executing your program.



```
CALL   STK  SP=3FF9    ADDR   ---INST----- OP          REGS
----   ------------    -------------------- --    ---------------
2AB5   2AB5=E604 ..    3000   CALL     CAB4 CD    AF=00FF MZAPC
2388   2388=0418 ..    3003   JMP      0040 C3    BC=03DB=CD18 ..
1234   1234=02B2 ..    3006   SBB   E       9B    DE=2310=8303 ..
       FF00=FFFF ..    3007   INR   C       0C    HL=2320=C00F ..
       FFFF=C3FF ..    3008   POP   H       E1

2000   MVI   A,02    3E             200D   SHLD     2330 22
2002   STA     2300 32              2010   SHLD     232F 22
2005   LXI   D,2310 11              2013   STA      2400 32
2008   STAX  D      12              2016   DCR   A       3D
2009   LXI   H,2320 21              2017   JNZ      2002 C2
200C   MOV   M,A    77              201A   MVI   B,02   06
                                                 COPYRIGHT 1977
=>GO,D000.                                     COMPUTER MART NJ
```

DDS display with DI (Display Instructions) command.

Since DDS executes about 300 instructions per second maximum, a standard breakpoint facility is available for running sections of code that are lengthy or critical about timing.

Evaluation

This is an extremely powerful software debugging tool. What more do you need for debugging programs? At all times you know the status of almost everything inside your computer and if your program has done something foolish. DDS might even be a handy tool for teaching the inner workings of the 8080 (provided it wasn't overwhelmed by DDS).

Anyway, I couldn't think of anything critical to say about DDS, so Larry Stein (owner of the Computer Mart of NJ) mentioned a few questions people have asked about the Dynamic Debugger.

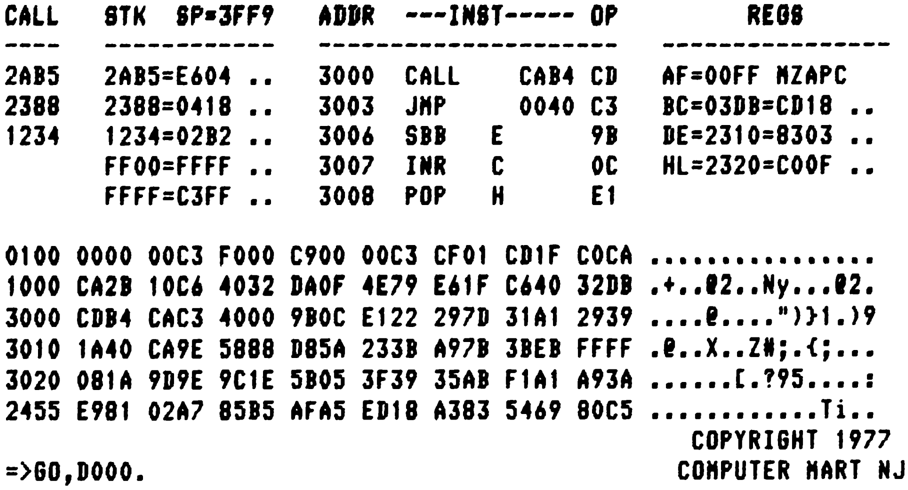
(1) Can it debug Z-80 programs? Well, sort of. When DDS reaches Z-80 code, it will display an II (Illegal Instruction) message and wait. You can then continue execution if you want. DDS doesn't display all the Z-80 registers, either. So, this version of the Dynamic Debugger really isn't designed for Z-80s. It's not for IBM-370's either, says Larry.

(2) What happens if you attempt to do an input from the keyboard which DDS is also using? Obviously that won't work out too well, because DDS is constantly checking the keyboard, since it stops whenever a key is pressed. The solution is to move into the accumulator what you were going to input. This is only a problem if you have only one keyboard in your system.

(3) What happens when your program executes code in ROM (for instance, an I/O routine)? While code in ROM is being executed, the DDS is inactive. It takes over as soon as control returns to code in RAM. If the code in ROM doesn't return to the instruction following the call, a breakpoint will have to be inserted. This shouldn't be much of a problem since programs in ROM are usually already debugged.

(4) How do you debug programs that use the VDM-1 for output? That might be a little tricky. The DDS does permit you not to refresh the screen after every instruction, and has a clear command which completely rewrites the screen. Useful if your program puts some junk on the screen. That's about the best you could ask for, since you're trying to debug and output with the same device. If you won't like it, you're welcome to your blinking lights!

Anyway, I consider most of these to be rather trivial questions.



```
CALL   STK  SP=3FF9    ADDR   ---INST----- OP          REGS
----   ------------    -------------------- --    ---------------
2AB5   2AB5=E604 ..    3000   CALL     CAB4 CD    AF=00FF MZAPC
2388   2388=0418 ..    3003   JMP      0040 C3    BC=03DB=CD18 ..
1234   1234=02B2 ..    3006   SBB   E       9B    DE=2310=8303 ..
       FF00=FFFF ..    3007   INR   C       0C    HL=2320=C00F ..
       FFFF=C3FF ..    3008   POP   H       E1

0100 0000 00C3 F000 C900 00C3 CF01 CD1F C0CA ................
1000 CA2B 10C6 4032 DA0F 4E79 E61F C640 32DB .+..@2..Ny...@2.
3000 CDB4 CAC3 4000 9B0C E122 297D 31A1 2939 ....@....")}1.)9
3010 1A40 CA9E 5888 D85A 233B A97B 3BEB FFFF .@..X..Z#;.{;...
3020 081A 9D9E 9C1E 5B05 3F39 35AB F1A1 A93A ......[.?95....:
2455 E981 02A7 85B5 AFA5 ED18 A383 5469 80C5 ............Ti..
                                                 COPYRIGHT 1977
=>GO,D000.                                     COMPUTER MART NJ
```

Dynamic Debugging System display with DM (Display Memory) option in use.

The DDS is available for $30 on CUTS or Tarbell cassette, and $35 on iCOM floppy disk, from the Computer Mart of New Jersey, 501 Route 27, Iselin, NJ 08830. It will be provided with a relocating loader (another nifty idea) which will let you stuff DDS anywhere in your system's memory. DDS is 4K in size. Complete details on interfacing to your system are provided. Despite my proclivity for finding problems in almost any product, this is one I can recommend without hesitation. Get one if you do any 8080 assembly-language programming, or just like to hack around with fancy software! ■

Dynamic Debugging System Commands

AR	Address range stop request
AS	Address stop request
BK	Run at full speed until breakpoint is reached
CALL	Pushes a word on DDS's CALL stack
CLR	Completely rewrites the screen
DI	Display memory as instructions in mnemonic format
DM	Display memory as hex and ASCII characters
DY	Set time delay between instructions
EB	Enter bytes into memory
EC	Enter characters into memory
EP	Enter program counter
ER	Enter register
ERP	Enter register pair
ES	Enter stack pointer
EW	Enter word into memory
FILL	Fill block of memory with a constant
FIND	Find a series of bytes in memory
GO	Same as BK except that the breakpoint is not retained
MOVE	Move blocks of memory
OS	Opcode stop request
POP	Pop a word from the stack
PUSH	Push a word on the stack
REF	Controls whether the DI or DM display is automatically updated
RET	Removes a word from the CALL stack
RS	Reference stop request
SR	Stack range stop request
SS	Store stop request
ST	Sets number of steps executed by stepping character
TC	Controls use of automatic return address validation
VSR	Value stop on register request
VSB	Value stop on memory byte request
VSP	Value stop of register pair request
VSW	Value stop of memory word request

Note: Most of these commands require arguments which are entered with the command, separated by commas. A period indicates the end of a command.

6800 Programming for Logic Design. Adam Osborne. Adam Osborne and Associates, Inc. 289 pp., paperback. $7.50. 1977.

With all the largely worthless books being published about microprocessors and microcomputers, it is refreshing to find one that sets its goals short of solving all the world's problems, and then meets these goals with a clarity, conciseness, and competence not often found in other books on the same subject. Author Osborne's intent is to show how assembly-language programs, in microprocessors, can replace discrete logic components. This is explained in the context of the Motorola 6800, but is applicable to almost any microprocessor. In seven chapters, the last two of which are 6800 reference material, the reader is guided from software simulation of gates to digital-system implementation via programming.

Osborne begins by showing how simple gates and flipflops can be replaced by small segments of assembly-language code. He then points out that using a microprocessor to replace a gate or a flip-flop is wasteful, and presents an example of replacing an entire circuit with a microprocessor. This circuit, used throughout the book, is an excellent vehicle for getting concepts across. The circuit is taken from an actual product, the Qume daisy-wheel printer, and provides a mixture of synchronization, timing delay, and control problems that manages to illustrate most of the major issues in this type of design. Again the reader is told that all is not perfect. The assembly-language program, while functionally correct, is merely a direct gate and flip-flop simulation of the print-wheel and hammer-control circuit. This is generally not the best approach as such direct simulation of a working digital circuit may not work due to unsuspected timing dependencies, lack of parallelism, etc. The circuit is then reimplemented, ignoring the gates and, instead concentrating on providing the functionality of the original circuit and preserving its input/output relationship. But the author is not finished yet. He describes a third, and final, program with some optimizations revealed by examining the problem from a software viewpoint.

A unique feature of this book is that it is printed in boldface and lightface type. The boldface type (more sparse) presents the key ideas, while the lightface type expands upon them. Thus the reader can skip the detailed explanations when not required, but, upon reaching an unfamiliar topic, will find sufficient detail to answer most questions. The only criticism is that the diagrams tend to be somewhat cluttered. Particularly annoying is the practice of drawing arrows directly on assembly-language and diagram text to illustrate a point. This, however, is a minor problem in an otherwise fine book.

The author assumes a knowledge of basic microprocessor principles. In practice, someone with a background in elementary combinational and sequential circuits and some form of programming will be able to benefit from this book. *6800 Programming for Logic Design* is not for those who get their thrills by playing Spacewar, but anyone who designs microprocessor systems, or who is interested in doing so, will find this book interesting, informative, and useful.

Jeff Grossman
Cambridge, MA

The fellow who is a good sport has to lose to prove it.

The man who is too old to learn was probably always too old to learn.

AN EVALUATION OF THREE 8080 8K BASICS

Steve North

In choosing a BASIC interpreter or other sophisticated and possibly expensive piece of software for your personal computer, what factor should be kept foremost in mind? A great deal of attention is sometimes paid to timing comparisons. While the speed of an interpreter can be important in some cases, most personal-computer applications tend to be I/O-bound, which makes the internal speed of an interpreter of less significance. Just because one interpreter in 25% faster than another doesn't make it better. However, a well-written interpreter is probably faster than a poorly-written one. The time (or speed) that's really important in this evaluation is *yours*, not the computer's. If it takes you five hours more to code a particular program in one language than in another, thirty seconds hardly make a difference. What most people want is a language that permits them to do what they want with a maximum of flexibility and a minimum of effort (of course, this is only true if you program to do things and not program just to program). Factors such as memory size and availability of assembly-language source code will diminish in importance as memory gets cheaper and people who don't care about assembly language start buying computers. Reliability will become even more important since the average person finds bugs and crashes even more annoying than the computer fanatic does.

This issue we're bringing you an evaluation of three 8080 8K BASICs—MITS 8K BASIC, IMSAI 8K BASIC, and BASIC ETC. In the next issue we'll try to get to some 6800 and maybe Z-80 software, in addition to more 8080 software products (including a micro-APL, a BASIC-like pseudocompiler, a FORTRAN compiler, and a very extended BASIC).

MITS 8K BASIC 4.0

Author: Microsoft.

Size: 6.2K with all functions retained.

Price and Availability: $250, on papertape or MITS cassette. $75 for owners of MITS systems. MITS, Inc., 2450 Alamo S.E., Albuquerque, N.M. 87106

Reliability: No problems noted.

Documentation: The user's documentation for all the MITS BASIC (4K, 8K, and Extended) are lumped together in one manual, forcing the user to separate all the information pertaining to a particular interpreter from useless information on other BASICs. The instructions are above average and include examples. The source code is not available.

Speed: Our original benchmark took 20 minutes to reach 34 and 52 minutes to reach 56. SIDES3 took 43 seconds.

Features:

Commands: CLEAR (clears variables; with an argument, sets amount of string space), CLOAD (loads or checks file on cassette, may also be used as a program statement or command to load arrays from the cassette), CONT (continue after a STOP or control-C), CSAVE (saves a program file or array on cassette), LIST, NEW, NULL (sets number of null characters printed after a CR/LF), RUN.

Statements: DATA, DEF, DIM, END, FOR, GOTO, GOSUB, IF...GOTO, IF...THEN, INPUT, LET, NEXT, ON...GOSUB, ON...GOTO, OUT (outputs a byte to an I/O port), POKE (stores a byte in an absolute memory location), PRINT, READ, REM, RESTORE, RETURN, STOP, WAIT (wait for a specific value at an I/O port).

Variables: Variable names may be any number of characters in length, but only the first two characters are significant. Thus, MITS 8K BASIC doesn't know the difference between the variables DOG and DOUGHNUT, because they both begin with DO. Range is ±2.9387E-38 to ±1.70141E38.

Functions: +,-,*,/,↑, ABS, ASC (returns ASCII value of character string), ATN, CHR$ (converts from ASCII to character string), COS, EXP, FRE (number of free bytes in the system), INP (inputs a byte from an I/O port), INT, LEFT$ (takes left part of a character string), LEN, LOG, MID$ (middle part of a character string), OCT$ (converts a decimal number to a character string of octal digits), RND, POS (position of the print head), RIGHT$ (right part of a character string), SGN, SIN, SPACE$ (creates a string of spaces), SQR, STR$ (converts a number to a string of digits), TAB, TAN, USR, VAL (converts string of digits to a number).

User-defined Functions: Yes, with any number of arguments (such as DEF FNA(X, Y, Z)=X+Y*Z). String functions are also permitted (DEF FNZ$ (X$) =X$+"HELLO").

Arrays: Yes, with any number of dimensions.

Machine-Language subroutine interfacing: Yes, with the USR function. USR has one argument which the machine-language subroutine picks up by calling the subroutine whose address is stored in locations 4 and 5. The address called by USR is set up by POKE-ing memory locations 111 and 112 with the desired address. A value can be passed back to BASIC by placing it in two CPU registers and then calling another subroutine in BASIC.

Character Strings: MITS 8K BASIC has very complete character-string-handling features. Strings may be handled as scalars (X$="HELLO") or as arrays (X$(1,2)="HELLO), etc.) Strings may contain up to 256 characters. The plus symbol (+) is used for concatenation.

Formatted Print: None

Editing Functions: None.

External Files: Only to the extent of being able to save and load arrays from the cassette with a program statement.

Error Messages: 18, specified by a two-character error code.

Extra Stuff: Control-0 is used to control the printing of output on the terminal. "?" is equivalent to PRINT. Control-Q and Control-S are used to temporarily freeze and then continue execution of a program or listing. The logical functions AND, OR, and NOT are available. MITS BASIC has a initialization dialog that permits some functions to be dropped to release more memory for programs. Depending on the sense-switch setting at initialization time, MITS BASIC configures itself for any one of a variety of MITS I/O boards.

User Comment: MITS 8K BASIC was the first widely available microcomputer BASIC and it has remained an outstanding product, a standard with which to compare other BASICs. Unfortunately, its high price discourages its use by low-budget computer hobbyists. But if you can afford it, this is the way to go.

IMSAI 8K BASIC 1.4

Author: IMSAI

Price and Availability: $100 for a papertape or Tarbell cassette. IMSAI Manufacturing Corp., 14860 Wicks Blvd., San Leandro, California 94517.

Size: 8.8K

Documentation: The instructions are merely a list of 8K BASIC's statements and functions with a little explanation, and some random comments on using 8K BASIC. Quite often the instructions are insufficient; for instance, the CHANGE verb is explained by "CHANGE string to array, or CHANGE array to string." Since the DEC BASIC-Plus manual is listed as a reference, one is probably expected to obtain a copy to get a better idea of how to use IMSAI 8K BASIC. The manual includes 113 pages of well-commented source code for 8K BASIC in assembly language.

Reliability: IMSAI 8K BASIC flunks the notorious 10 GOSUB 10 test by recursing itself to death! It seems that IMSAI 8K BASIC never checks to see if it runs out of memory for the BASIC program, variables, or if the stack eats its way down through the BASIC program. In fact, it doesn't even have an out-of-memory error message! There are a few other problems—according to the manual, strings over 238 characters can "destroy the interpreter," and active FOR/NEXT loops may not be restarted. The floating-point math routines also need some help (did you know that 2↑200 is -1.38771E-17? Computers *never* lie!)

Speed: AHLDIG: 89 minutes to reach 34, 160 minutes to reach 56. SIDES3: 103 seconds.

Features:

Commands: CON (continues execution after a STOP or control-C), FRE (prints number of free bytes in system), KEY (used to return to normal mode after a TAPE command), LIST, NEW (an optional form of this command deletes the program but not variables), RUN, SAVE (punches papertape of program), TAPE (loads papertape of program without echoing on the terminal), XEQ (runs program without clearing variables first).

Statements: CALL, CHANGE (convert from string to array or vice versa), DATA, DEF, DIM, END, FOR, GOSUB, GOTO, IF...THEN, IF...GOSUB, IF...GOTO, INPUT, INPUT LINE (used to input a character string with embedded quotes or commas). NEXT, ON...GOSUB, ON...GOTO, OUT (outputs a byte to an I/O port), POKE (stores a byte in an absolute memory location), PRINT, RANDOMIZE (used to true pseudorandom numbers), READ, REM, RESTORE, STOP.

Variables: A-Z, AO-Z9. Range is ±2.7105E-20 to ±5.7646E18.

Functions: + , -, *,/,↑, ABS, ASCII (returns the ASCII value of a character string), ATN, CHR$ (converts from ASCII to a character string), COS, EXP, INP (inputs a byte from an I/O port), INSTR (used to determine the position of one character string within another), INT, LEFT$ (takes left part of a character string), LEN, LN (natural log), LOG (log base ten), MID$ (middle part of a character string), NUM$ (converts a string of digits into a number), PEEK (returns the value of an absolute memory location), PI (3.14159), POS (returns the position of the print head), RIGHT$ (right part of a character string), RND (without an argument works like normal RND; with an argument, returns a value between one and the argument), SGN, SIN, SPACE$ (creates a string filled with spaces), SQR, STRING$ (same as CHR$, but creates multiple-character strings), TAB, TAN, VAL (converts a number to a string of digits).

User-defined functions: Yes, only one argument is allowed.

Arrays: Arrays may be named A-Z and have one or two dimensions.

Machine-language subroutine interfacing: Yes, but there is no direct way to pass values between BASIC and the subroutine (except by PEEK and POKE).

Character Strings: Strings may contain up to 256 characters, but string arrays are not permitted. The plus symbol (+) is used for string concatenation. Note that the manual reports that there is a bug which causes the interpreter to bomb if strings greater than 238 characters are used. IMSAI 8K BASIC has a complete set of string-handling functions:

Formatted Print: None.

Editing Functions: None.

External Files: None.

Error Messages: 10, indicated by a two-character error code. Overflow and underflow errors are non-fatal (execution of the program continues after a message is printed).

Extra Stuff: "?" and "!" may be substituted for PRINT and REM. Print statements may use single or double quotes for delimiting literal strings. Control-0 is used to control the printing of output. The NEW and XEQ commands provide a rudimentary facility for chaining programs together.

IMSAI also has a expanded version of 8K BASIC 1.4, called 9K BASIC 1.4. This version of BASIC includes CSAVE and CLOAD (cassette save and load of BASIC program) for the Tarbell cassette interface on an IMSAI MIO board, an EDIT command (of the form EDIT (line number) (delimiter) (old text) (delimiter) (new text)), and a BAUD command used to change the terminal baud rate on an MIO board.

User Comment: The apparent intent in writing IMSAI 8K BASIC was to make it as close to DEC BASIC – Plus as possible. In fact, 12K BASIC was supposed to be completely compatible with BASIC-Plus. But alas, along the way, IMSAI 8K BASIC got derailed. There are still some serious problems that need to be ferretted out before IMSAI 8K BASIC can compare with the quality of IMSAI's hardware products. But we do like the idea, and hope that IMSAI will finally clean up this interpreter. Incidentally, PEEK, POKE, CALL, INP, and OUT are not BASIC-Plus syntax—more like MITS BASIC. Obviously, what's great for some computers may need some alteration for micros. One other comment: we remember reading in IMSAI ads that 4K, 8K, and 12K BASIC would be free to all registered IMSAI owners. Now the price is $100. In its present form, MITS 8K BASIC at $250 seems like a better buy.

BASIC ETC.

Authors: John Whipple and Dick Arnold.
Size: 9K
Price and Availability: On papertape, Kansas City or Suding cassette for $25. Binary Systems, Inc., 6345 Central Expressway, Richardson, 75080.
Documentation: The 30-page instruction manual includes examples and is generally satisfactory. Details on customization are included. The source code is not available.
Reliability: BASIC ETC flunked the 10 GOSUB 10 and the 10 RETURN tests. It also inserted its own line in our benchmark program when a variable overflowed. In none of these cases did the interpreter actually blow up, but it did damage the BASIC program.
Speed: AHLDIG: In real mode, 39 minutes to reach 34, and 100 minutes to reach 56 (the end of the test). In integer mode, about 24 minutes to reach 34. SIDES3 could not be run, because there are no trig functions.
Features:

Commands: LIST, LOAD (loads program and variables (optional) from cassette), NEW, NULL, (sets number of null characters printed after a CR/LF), RUN, SAVE (save program and variables (optional) on cassette).

Statements: DATA, DIM, END, FOR, GOSUB (with an optional list of expressions passed to the subroutine), GOTO, IF, INPUT, LET, MEMW (stores a byte in an absolute memory location), NEXT, OUT (outputs a byte to an I/O port), PRINT, PRMT (sets prompt character used in the INPUT statement), READ, REM, RESTORE, RETURN, SD (sets number of significant digits, 6-72, used internally), STOP, S$SPC (sets amount of string space), VAR (used by a subroutine to pick up arguments passed to it by the calling program), ZONE (sets width of numeric print zone).

Variables: A-Z are integer variables (0 to ± 32767). A0-Z9 are real variables (1E-62 to 9.99...E62).

Functions: +,-,*,/, ABS, FLT (converts from integer to real), FRE (returns the number of free bytes in the system), IN (inputs from an I/O port), INT (converts from real to integer), LEN, MEMR (returns the value of an absolute memory location), RND, SQR, USR, VAL (converts a string of digits to a number).

User-defined functions: Yes, but not stardard BASIC. User-defined functions are numbered FN0 through FN7 and may have one argument that's represented in the function definition by a pound sign (#). For instance, DEF FNZ(X)=27*X in normal BASIC is written as FN2=27*# in BASIC ETC. Strange!

Arrays: Multidimensional arrays are supported. However, only real arrays are permitted and they must have real variable names. Thus, DIM Q(10) is illegal but DIM Q7(N) is OK. The subscripts must be integer mode expressions.

Machine-language subroutine interfacing: Provided by the USR function. Arguments are passed both ways with two CPU registers.

Character Strings: Yes, but they are merely alternate representations of real variables, not distinct. Thus A0 and A0$, and B9(1,2) and B9$(1,2) actually represent the same data. This eliminates the need for string-handling functions. The plus symbol (+) is used to concatenate strings. There is a provision for taking substrings in BASIC ETC. Obviously, not standard BASIC.

Formatted Print: The ZONE command can be used to set the number of positions in the numeric PRINT zone. BASIC ETC also has a non-standard form of PRINT-USING with integer, real, exponential, ane ASCII specifications.

Editing Functions: None.

External Files: None.

Error Messages: 27, specified by a three-digit number (100 to 126).

Extra Stuff: None.

User Comment: BASIC ETC is cheap and it does work, but we don't care for some of its non-standard syntax. For instance, there is no THEN in an IF statement! To write an IF... THEN, you use the colon (multiple-statement separator) in place of THEN, as in IF X=1: PRINT "HI", which probably saved someone three or four bytes in writing BASIC ETC. But the really annoying thing is that BASIC ETC distinguishes between integer and real data types, and you're not allowed to say for yourself if you want X5 to be an integer or a real number. True, integer math in BASIC ETC in significantly faster than real math, but this isn't standard BASIC. Mixing of integer-data types with real-data types is not permitted (so PRINT 2*X5 bombs, because BASIC ETC decides that the expression must be an integer since 2 is an integer). Further, there is no actual INT function in BASIC ETC. To get the integer part of a real variable you can always resort to X5=FLT(INT(X5)), which only works for certain values of X5. We could go on, but you get the idea. BASIC ETC is not, in some areas, anything like any other BASIC we've used. BASIC ETC *is* the most inexpensive floating-point BASIC we've tested so far. ■

New Benchmark Program

Geoffrey Chase

We're introducing a new benchmark program, SIDES3, written by Geoffrey Chase. (He also sent us a nifty Heapsort benchmark which we didn't have time to use yet). SIDES3 will be used primarily to test the speed of a language's function library, on the assumption that if the trig functions are fast, then the rest of the function library also is, since programmers are usually consistently tricky or conventional. Geoff reports the following results:

EDU-20 BASIC, one-user configuration	197	seconds
OS/8 FORTRAN IV	8.8	seconds
FORTRAN-2 with E.A.E. hardware	.27	seconds
U/W FOCAL Version R (carrying 10-place accuracy)	32	seconds

```
SIDES3  3/30/77  EDUCOMP BASIC V2.0

 100 REM.       Program to solve S.S.S. (given 3 sides of a triangle)
 110 !          PAS 3/77        for benchmarking
 112 INPUT A$
 115 FOR I=1 TO 100                ! benchmark -- otherwise very stupid !
 120 RESTORE
 125 READ A,B,C
 130 IF C>=A THEN 170
 140 T=C
 150 C=A
 160 A=T                           ! SWAP 'EM
 170 IF C>=B THEN 210
 180 T=C
 190 C=B
 200 B=T
 210 REM.       USE LAW OF COSINES
 220 U=C*C - A*A - B*B
 230 U=U/(-2*A*B)                  ! U = COS(ANGLE 'C') opp. to side 'c'
 240 IF U=0 THEN 300               ! Beware of 90 DEGR. (PI/2)
 250 U=SQR(1-U*U)/U                ! U = TAN(ANGLE 'C')
 260 U=ATN(U)                      ! U = ANGLE opposite to side 'c'
 270 IF U>=0 THEN 310
 280 U=U+PI                        ! correct the quadrant
 290 GOTO 310
 300 U=PI/2                        ! this 'BASIC' keeps a permanent 'PI'
 310 A1=U
 320 R=SIN(U)/C                    ! sin/opp. side: constant for triangle
 330 REM.       Now look at side 'a' and opp. angle 'A'
 340 U=A*R                         ! U=SIN(A)
 350 GOSUB 530
 360 A2=U
 370 U=B*R
 380 GOSUB 530
 390 A3=U
 395 NEXT I                        ! end of benchmark
 400 PRINT
 410 PRINT "SIDE A =";A;" ","SIDE B =";B;" ","SIDE C =";C
 420 PRINT
 430 PRINT "ANGLE A",A2;"RADIANS",
 440 U=A2
 450 GOSUB 600
 460 PRINT "ANGLE B",A3;"RADIANS",
 470 U=A3
 480 GOSUB 600
 490 PRINT "ANGLE C",A1;"RADIANS",
 500 U=A1
 510 GOSUB 600
 520 GOTO 700
 530 IF U=1 THEN 570               ! Beware of 90 DEGR. (PI/2)
 540 U=U/SQR(1-U*U)                ! U becomes tangent
 550 U=ATN(U)
 560 GOTO 580
 570 U=PI/2
 580 RETURN
 590 !  ------
 600 U=U*180/PI                    ! to degrees
 610 D=INT(U)
 620 U=U-D
 630 U=U*60
 640 M=INT(U)                      ! minutes
 650 U=U-M
 660 U=U*60
 670 S=INT(U+.5)                   ! seconds
 680 PRINT D;"degr.";M;"min.";S;"sec."
 690 RETURN
 695 DATA 120, 207.846, 120
 700 END
```

```
SIDES3  3/30/77  EDUCOMP BASIC V2.0

?

SIDE A = 120                  SIDE B = 120                  SIDE C = 207.846

ANGLE A       .5236 RADIANS 30 degr. 0 min. 0 sec.
ANGLE B       .5236 RADIANS 30 degr. 0 min. 0 sec.
ANGLE C       2.09439 RADIANS               119 degr. 59 min. 60 sec.

READY

AROUND 6.6 SECONDS.......(UNTIL PRINTING BEGAN)         ^U
```

Program Notes

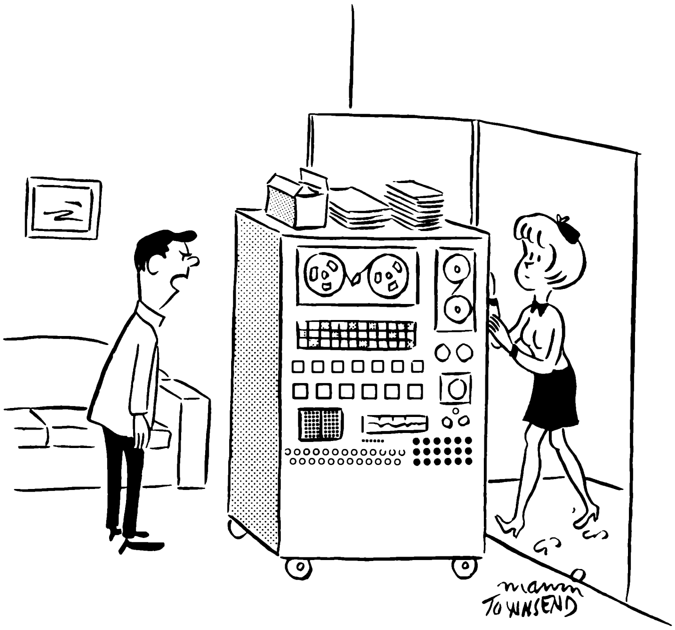
Why swap A,B,C?

(a) This is inherited from a test program that did a run-time INPUT. (Kill 112 through 125, write "120 INPUT A,B,C." Also kill 395 and 695, natch.)

(b) The arc cos (effective) is sensitive to quadrant. If C is the largest side, the opposing angle is the only possible obtuse angle and gets trapped rather neatly. Otherwise one would have to move cautiously with the (effective) arc sin — the value returned might need replacement by 180° value, or not. This gets rather nasty.

(c) I have versions of the A.K. Head recursive arc sin and arc cos for BASIC. Slow, but they can be useful. No need to worry about (2h-1)-90°! ■

JOE'S BAR
SNOBOL SPOKEN HERE



"Didn't I warn you it would lead to this? First your boss asks you to bring home a little book work, then typing, then a few reports to write up...!"

Two Space Games (With Graphics!) For Your Home Computer

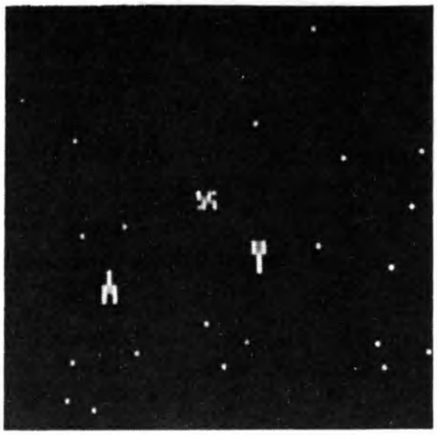
Steve North

Do you own a Cromemco TV Dazzler or a Processor Technology VDM-1? If not, and if you're a fancier of space games, you might want to get one (or both)! Space games for personal computers with graphics have arrived, in the form of Cromemco's SPACEWAR and Processor Tech's TREK 80. These games offer you sophistication you won't find in any coin operated machines.

SPACEWAR

Of the two games, SPACEWAR provides more impressive graphics—because the TV Dazzler was designed with graphics in mind, while the VDM-1 does a better job of displaying characters. SPACEWAR is modeled closely on the old Spacewar game which has been around for years. It's played on a TV screen which represents the galaxy. However the galaxy isn't planar. If you drift off one side of the screen, you reappear on the other side, so actually the galaxy is a sphere. The four corners of the screen represent the same point, the point furthest from the center. The objects in the galaxy are the spaceships, torpedoes, stars, and the sun.

Within this portion of space, the two combatant's spaceships travel around a central sun and are attracted to it by gravity. The spaceships are controlled by Cromemco JS-1 joystick consoles. Push the stick forward, and your ship accelerates. Push the joystick to the right or left and your ship rotates clockwise or counterclockwise. Push button one on the joystick console, and a torpedo is fired from the nose of your ship in a direction relative to that of the vessel. The object of the game is to blow up your opponent's ship with a torpedo, while your ship remains intact. You have 32 torpedos and 30 seconds worth of fuel. Torpedos self-destruct after a short period. Their range is thus limited by their speed. If button 1 is held down, torpedoes will be fired in a machine-gun-like fashion at the rate of 2 per second. Pull the stick all the way back, and you enter hyperspace. After a few seconds your vessel pops out of hyperspace in a random location, but disguised as a star. After a few more seconds, the star turns back into your vessel, with a random course and speed. So what's to prevent you from hyperspacing continuously? First, a timer prevents you from re-entering hyperspace immediately after exiting. Second, when you exit hypespace, there is a 1/8 chance of blowing up. Third, while your ship is a star it is vulnerable to attack. Hyperspace is best used only as a last resort to escape an opponent's torpedo which can't be shot down.



SPACEWAR display, with whirling swastika-like sun at center, toward which spaceships are attracted by gravity.

As mentioned before, there is some additional scenery in this game. The slowly rotating star field basically functions as background, probably to hide ships when they pop out of hyperspace. The sun, at the center of the screen, has a gravitational effect which can make for some interesting maneuvers. If your ship falls in, it gets split up into all four corners of the screen (which isn't fatal, since they represent the same point). By setting console switches at the start of play, you can eliminate the starfield, eliminate the sun, or make the sun lethal, which means that if you fall in, you lose! (This is preferred for "serious" play.)

TREK 80

TREK 80, as all you superbrains will have guessed, is based on Star Trek. Actually TREK 80 has borrowings from both the standard matrix-oriented Star Trek game (such as Super Star Trek) and TREK 73, a rather sophisticated pseudo-realtime Star Trek in HP 2000 BASIC (no, don't write for copies, please).

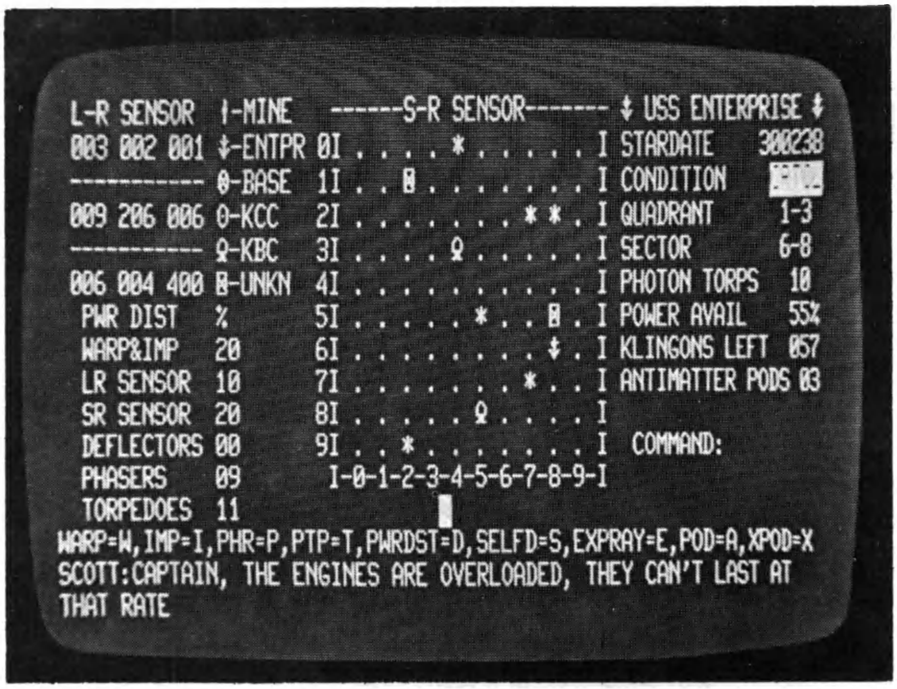
TREK 80 is a real-time game, as opposed to most Star Treks in BASIC which let you enter a command, execute it, stop all the action and get another command, etc. In TREK 80 things are happening while you type in commands. It's played in 10-by-10 galaxy sectors, with the following objects: the Enterprise (you), Klingons, space mines (which explode when disturbed), starbases, and unknown objects. All the data needed for playing TREK 80 is displayed on the VDM constantly. The short-range scan is placed at the center of the screen, with the long-range scan in the upper left, status report in the upper right, and miscellaneous messages displayed in the extreme bottom section of the screen. Also on the screen are a key to the short-range scan, a list of commands, and a place for you to type in commands. All these displays are updated constantly, so that as you're warping through quadrants, you see them on your short-range scan and the adjacent quadrants on the long-range scan. Here's a brief rundown on the TREK 80 commands:

Warp engines: As in regular Star Trek, they are used to move you from quadrant to quadrant. But in TREK 80 you merely specify the course, not warp factor. The ship then moves at a constant speed in the designated direction, and stops when you enter a new command.

Impulse engines: Basically the same as warp engines, except that they move you only within your present quadrant and shut down upon entry into an adjacent quadrant.

Phasers: You enter the direction of phaser fire and then they zap whatever is in the way.

Torpedos: Same as phasers except



```
L-R SENSOR  !-MINE    ------S-R SENSOR------- ‡ USS ENTERPRISE ‡
003 002 001 ‡-ENTPR 0I . . . . * . . . . . I STARDATE    300238
----------- Θ-BASE  1I . . Θ . . . . . . . I CONDITION   [illegible]
009 206 006 0-KCC   2I . . . . . . . * * . I QUADRANT     1-3
----------- ♀-KBC   3I . . . . ♀ . . . . . I SECTOR       6-8
006 004 400 ▣-UNKN  4I . . . . . . . . . . I PHOTON TORPS  10
 PWR DIST   %       5I . . . . . * . . ▣ . I POWER AVAIL  55%
 WARP&IMP   20      6I . . . . . . . . ‡ . I KLINGONS LEFT 057
 LR SENSOR  10      7I . . . . . . . * . . I ANTIMATTER PODS 03
 SR SENSOR  20      8I . . . . . ♀ . . . . I
 DEFLECTORS 00      9I . . * . . . . . . . I  COMMAND:
 PHASERS    09        I-0-1-2-3-4-5-6-7-8-9-I
 TORPEDOES  11                █
WARP=W,IMP=I,PHR=P,PTP=T,PWRDST=D,SELFD=S,EXPRAY=E,POD=A,XPOD=X
SCOTT:CAPTAIN, THE ENGINES ARE OVERLOADED, THEY CAN'T LAST AT
THAT RATE
```

TREK 80 display, with short-range scan at center

more powerful.

Energy distribution: You are permited to alter the percentage of energy allocated to any of the devices on the ship. Our use of this command has not been entirely successful because the ship blows up almost every time we use it.

Experimental ray: A very interesting feature, because you never know what the experimental ray is going to do. It can make the Klingons invisible, destroy all the ones in your quadrant, freeze time for the Klingons, jumble the quadrant, or disturb the space in your quadrant. It can also cause a "computer malfunction," which means it puts the system in an infinite loop!

Pods: There are two commands for control of antimatter pods, A and X. They seem to refer to launching pods and exploding them respectively.

A Brief (Yet Useless?) Comparison

Why do we say useless? Because who in their right mind is going to buy $200 to $500 worth of hardware to play a $15 game? But as long as we're beating this thing to death....

Obviously SPACEWAR is the more visually striking game of the two. However it does predicate that you have more of an investment in hardware then TREK 80 (a TV Dazzler, A/D, and two joystick consoles, vs. a VDM-1). Although the rules for SPACEWAR seem simpler than those of TREK 80, it is a bit difficult to master the controls, since what you control is your acceleration, not velocity or position. Of course, that's probably more realistic, and certainly more challenging. Because of the graphic capabilities of SPACEWAR, you can try out some interesting tactics—like sending torpedoes off one end of the screen, so they will appear from nowhere on the other side; or using the sun for a slingshot effect to save fuel. TREK 80 is a much more complex game (from the user's viewpoint—offhand, the idea of programming SPACEWAR in machine code seems a little overwhelming).

It might be helpful at this point to note that our experience with playing TREK 80 was a little, uh, unique. We tried it out on a newly acquired Sol System, courtesy of Processor Technology. Unfortunately the Sol temporarily had the wrong character generator ROM in its VDM. The symbolic character generator displays control characters as special freaky symbols such as backwards question marks, check marks, little lightning-bolt-shaped characters, etc. These symbols are used by TREK 80 to represent the objects in the short-range scan. Our VDM displayed the control characters as infinitesimally small two-letter abbreviations, so it was almost impossible to tell who was what on the short-range scans. We also did not have the instructions to TREK 80, because as of this writing they don't exist (though Processor Tech will give you the instructions when you buy the game). Consequently our style of play was rather strange. While I would have preferred a more rational, conservative approach, Dave Ahl seemed bent on blowing up as many objects as possible without regard for what they were. (We got the correct character generator a few days later and it enhanced play considerably.)

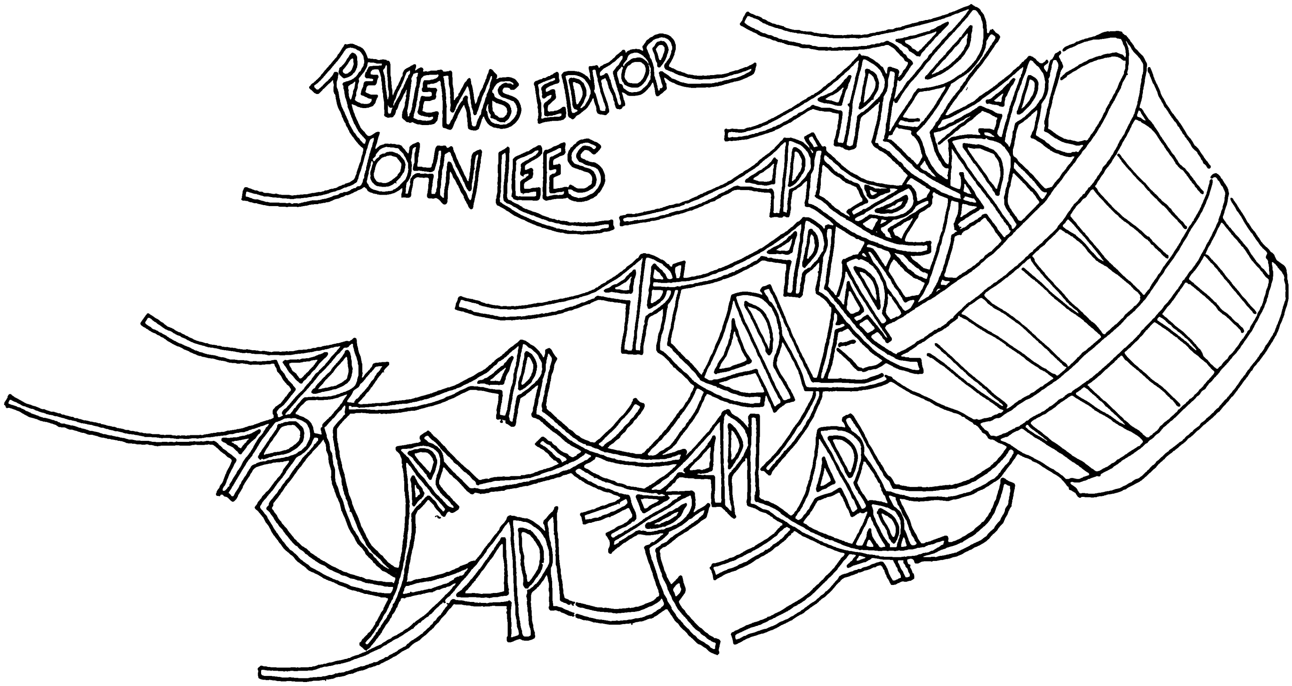
If you're the kind of person who is easily bored by games like Pong, you may find the same to be true of SPACEWAR after a while (although we doubt it). Of course not everyone likes Star Trek games either, so you'll have to decide for yourself. One feature of TREK 80 we liked was that, at the start of the game, you are permitted to set the speed of the simulation. SPACEWAR is always played at the same speed—quite possibly because it is a very complex simulation and the system may be computing at full speed. SPACEWAR has to move all the objects in the galaxy around at the same time, and check to see if any are getting too close, since torpedoes can detonate each other. (This type of problem is also found in TREK 73, the simulation-type Star Trek game mentioned before, which really bogs down when many objects have to be moved around.) At maximum speed in SPACEWAR, your vessel can cross the screen in about one second. However this can only be done by first constant burning in one direction for thirty seconds. The controls do seem a little sluggish, which may also contribute to the difficulty in learning to operate them. This criticism of SPACEWAR should not be interpreted to mean that it's not a fantastic game, though! After a few hundred hours we can vouch to it's addictive effect. Besides, microcomputers are not noted for their high computing speed. One thing about TREK 80—all the courses are in integers, 0 to 7. Is that bad? Maybe not. It's almost quicker to just move up a sector or two, than it is to try to figure, "Is that going to be a 5.6 or a 5.7?" TREK 80 apparently already uses all the graphical capabilities of the VDM, while SPACEWAR is a black-and-white game which could be more exciting if color were used. Imagine, one ship green, the other blue, yellow sun and exhaust, white stars, red torpedoes, big colorful explosions when someone gets hit! The truth is that though you can think of improvements for almost any game, these games are fascinating as they are now—our hats are off to the people who programmed them, *in assembly language!* And, not to show any kind of partiality, if any of the manufacturers of other video devices (such as Merlin, etc.) have any interesting graphic games, we'd be happy to try them out.

Sources

SPACEWAR is available on paper tape with complete documentation for $15.00 from Cromemco, 2432 Charleston Road, Mountain View, CA 94043. (415) 964-7400.

TREK 80 is available on cassette tape for $9.50, or on paper tape for $14.50; both with complete documentation, from Processor Technology Corp., 6200 Hollis Street, Emeryville, CA 94608. (415) 652-8080. ■

A Bushel of APL



The Reviews Editor gave me a bundle of publications on APL all at once, so it seems proper to do a composite review. But first it might be appropriate to introduce APL to those who are not familiar with it. So far it has not swept the microcomputer field, but the entry has begun, and APL has generated as much heat and argument as any computer language. It has raised antagonism and won fanatic devotees and we should at least be aware of APL's existence.

APL began with Kenneth E. Iverson's book *A Programming Language* (Wiley, 1962). Hence the name APL. So many of the hot topics of today — microprogramming, stack computers, associative memories — are covered in that book that one despairs how slow, not how fast, progress in computing is. Iverson then began working with Adin Falkoff at IBM to implement the language. In that process the syntax of APL was modified to its current form to make it more suitable for computer use and implementation. In the meantime APL was used in publishing the formal definition and description of the then-new IBM System/360 series of computers in 1964. In 1966 the first implementation, *APL/360*, became available to users.

By design, APL is a language to be used from an interactive terminal. Editing, tracing, error messages, and other debugging facilities are an integral part of the language, not *ad hoc* facilities added more or less well, and differently, to each implementation. It is also a language inherently suited to run-time interpretation, rather than compilation and subsequent execution of compiled code. Indeed, it is generally claimed that APL cannot be compiled at all. However, it can benefit greatly from the inclusion of special facilities in the microprograms of host computers as, for instance, is the case with the *APL ASSIST* feature on later models of the IBM System/370. I predict that soon you will be able to buy off-the-shelf ROMs for most microcomputers preprogrammed with APL interpreters. But you will also need up to 64K bytes of working storage and a terminal with the APL symbol set. This latter point has some people upset, APL uses its own symbol set which has over 60 special characters in addition to the alphabet and digits. Some of the special characters are overstrikes, composites of two symbols. While implementations of APL that allow the use of ordinary ASCII terminals exist, I cannot imagine using it this way. In ASCII the special APL symbols have to be represented by the dollar sign trailed by two more or less mnemonic characters, and the whole flavor of the language, and the concise and clear statement form, are lost.

The "natural" data item of APL is the array, of any rank. Scalars, of rank 0, are a special case and sometimes behave rather differently from what would seem to be the same, a vector of one element only. The basic scalar data types of which arrays may be composed are numbers (some implementations distinguish between integers and floating-point numbers internally, but transfer functions are applied automatically), characters, and logical values. The scalar functions of APL are so called not because they only take scalars as arguments but because they operate on arrays element by element, while the mixed functions may operate on arrays as a whole. All APL functions defined by the user are potentially recursive. They may have local variables and the scope rules for these are dynamic, not the static scope rules of block-structured languages. If you think about it, this is a necessity with the interpretive nature of APL. The "unit" of APL is not a program, as in other languages, but the workspace. This may contain both variables and user-defined functions. A workspace may be filed and retrieved without harm to the values of its variables; indeed, it is possible to suspend function execution, save the workspace, and later retrieve it and resume execution.

Currently, APL is available on most timesharing systems and one worldwide service. I. P. Sharp & Associates Ltd. and Scientific Timesharing Corporation offer no other language on their bureau machines. Among IBM (US) employees there are reputed to be 28,000 active users of the language. It is taught at many schools and colleges. I know of one program in which all students at a nearby junior high school learned APL for some years, until the school board cut the funds for computer access!

Now to the books.

★

APL An Interactive Approach. Leonard Gilman and Allen J. Rose. John Wiley & Sons Inc. 378 pp, paperback. $11.95. Second edition revised, 1976.

Introduction to APL and Computer Programming. Edward Harms and Michael P. Zabinski. John Wiley & Sons Inc. 400 pp, paperback. $10.95. 1977.

☆

Both of these books are addressed to the beginning student with no prior knowledge of computers or APL, or mathematics. But they also serve well as teaching manuals for those who have some previous experience with programming and wish to learn APL. At first sight this may seem an insult to the experienced programmer. These, and most other, APL texts are so full of seemingly trivial and repetitious exercises that they could not possibly serve other than the novice. But that is the point, the novice at APL. All of us carry a burden of assumptions from our first introduction to mathematics in school. Expressions are evaluated from left to right and there is a divinely ordained order of precedence amongst the arithmentic operators, or functions. APL violates both these comfortable illusions. In APL all expressions are evaluated right to left, and there is no precedence of operators other than that imposed by parentheses. The novice must shed previous assumptions, and the drill is the way to do that. Hence both texts assume that the student has ready access to an APL terminal and system to complete the exercises and to experiment.

It is difficult to choose between the books; any choice is probably a matter of personal taste. We do have some years' experience with Gilman and Rose with our own students, and they have no complaints about it. The other book is too new to assess that way.

While Gilman and Rose tend to introduce examples of APL applications as soon as the necessary functions and operators have been explained, the Harms and Zabinski book is in two parts, the first defining the language and the second giving examples of its application. The examples are also much better indexed. Both books cover examples ranging from mathematics thru scientific and engineering applications to data processing and commerce. Users of implementations on other than IBM systems will need their particular manuals in addition to either book to cover local log-on procedures, mass-storage facilities, and some tracing and debugging operations that relate to the system, and for other local system features.

The student exercises in both books are very comprehensive but where Gilman and Rose offer solutions to all problems, Harms and Zabinski provide only solutions to every second exercise.

★

The rest of the items to be reviewed are all from APL Press, Box 378, Pleasantville, NY, 10570. Payment must accompany any order, except for schools and libraries, but unlike most publishers APL Press offers quantity-purchase discounts to private individuals, on the following scale of total single order value: over $30 - 15%; over $100 - 20%; over $300 - 25%; over $1000 - 30%.

ALGEBRA: An Algorithmic Treatment. Kenneth E. Iverson. Addison-Wesley Publishing Co. and APL Press. 361 pp, paperback. $9.35. 1972.

Solutions to Iverson's Algebra. Janet A. Iverson. APL Press. 42 pp, paperback. $1.50. 1976.

Elementary Analysis. Kenneth E. Iverson. APL Press. 218 pp, paperback. $6.25. 1976.

CALCULUS in a new key. D. L. Orth. APL Press. 286 pp, paperback. $8.00. 1976.

☆

The three texts (and one manual of solutions to problems) offer an introductory course in mathematics at the high-school and college level. Alternatively, anyone reasonably confident in mathematics can use them to learn APL, although one of the texts reviewed earlier would probably be better for that.

The three texts form a series in the order listed above. The student is introduced simultaneously to algebra and to APL in the first book by Iverson. Teachers, and probably students too, should first read the two excellent appendices on "Algebra as a Language" and "Use of the Computer in Teaching." The first explains Iverson's view of mathematics and of the importance of particular mathematical notations in explaining, or obscuring, problems, and the second outlines how the computer can be used by the teacher to enhance comprehension of material in many subject areas. If, that is, the computer can be programmed in a pedagogically useful way. While access to an APL system is not essential, any course based on these texts should really be planned on the basis of regular use of the computer to complete the problems presented.

The mathematical style and approach of this series may be offensive to an older generation of mathematicians for whom vulgar computation, the generation of results, is best left to clerks while they contemplate the elegance of their formulations. But it turns out that APL notation does lead to elegant mathematics and is very helpful in suggesting solutions at the same time as it ensures rigorous expression. It also aids comprehension. My son is in Grade 7, and although he would still rather play football than read algebra, he can yet already make sense of the early chapters.

The other two volumes continue on from the introduction to algebra; here there is of course less emphasis on APL as a language and more on the mathematical treatment. Whether the approach chosen here is better than the conventional one cannot be judged out of thin air, it would require the opinion of teachers who have used the older approach and then these texts, after becoming sufficiently familiar and skilled in APL.

★

APL in Exposition. Kenneth E. Iverson. APL Press. 61 pp, paperback. $1.00. 1976

Introducing APL TO Teachers. Kenneth E. Iverson. APL Press. 25 pp, paperback. 75¢. 1976.

An Introduction to APL for Scientists and Engineers. Kenneth E. Iverson. APL Press. 26 pp, paperback. 75¢. 1976.

☆

These three booklets were originally published as IBM Technical Reports in 1972 and 1973. They are aimed at introducing teachers and members of the scientific and engineering community to APL.

APL in Exposition gives a basic introduction to APL and then devotes 6 to 10 pages each to illustrations of the use of APL in teaching various topics ranging from coordinate geometry thru logic to electric circuits. For computer buffs the most interesting things are a complete simulator for an 8-bit-wordsize, 8-instruction, 32-word-memory computer in two APL functions comprising 14 lines of code. An assembler for a symbolic assembly language for this machine to translate mnemonic opcodes and symbolic addresses to binary is given in seven functions totalling 19 lines. Or, if you would rather have a simulator that executes simple APL functions directly, that takes a function of a mere nine lines, but needs 19 auxiliary functions of 30 lines in all to transform APL statements into its required internal representation. By now you might have guessed that APL is indeed a very concise language! User-defined functions of but a single line, doing useful work, are common, even tho APL users have outgrown their early childish delight in the "one liner," the lust to write as complex and powerful a statement as the language will allow, without regard to comprehensibility.

Introducing APL... is largely a set of exercises, intended to be completed at the terminal in 15 to 20-minute sessions, five to six hours in all, for teachers who wish in turn to introduce their students to the computer and APL. The exercises come in sets, each on a particular topic, designed as a terminal session per set. The first one or two exercises of a set come with solutions to reinforce the student's confidence, and the next one or two draw out the user's skill in applying the knowledge already gained. The topics covered are those of interest in teaching mathematics and science in high school, graphs and power functions and statistical measures and the like.

An Introduction to APL. . . builds on a knowledge of vector algebra to teach APL and the use of the computer in scientific and engineering applications. Somehow this booklet lacks the sparkle and enthusiasm of the others. It is a potboiler, with examples too pedestrian and narrow in scope. Or mayhap that is a reflection of the author's estimation of that particular audience! But it still conveys the message that a large part of learning APL, and the potentialities of the computer, is to experiment, to play, at the terminal.

★

APL Reference Card. APL Press. 25¢. 1976.

☆

This vestpocket-size plastic reference card lists and gives examples of use of all the APL scalar and mixed functions, operators, system variables, and system functions. To say that it is a most useful tool to refresh the memory when sitting at a terminal, or jotting down a function, is to miss the point completely. Here we have what must be one of the most powerful and complex programming languages in existence, yet its essential points can be compressed onto the two sides of a 9.5-cm by 6-cm plastic card. So much for the vaunted orthogonality of ALGOL 68.

★

APL News. APL Press. 8 pp per issue.

☆

This is a periodic newsletter available on demand from APL Press. It publicizes their publications, offers space for interested users to explain novel applications or suggest improvements to the language, publishes APL games and puzzles, explains interesting techniques, gives notices of relevant meetings, etc., etc.

H.D. Baecker
Calgary, Canada ■

A Creative Computing Group Review . . .

PROGRAMMING PROBLEM BOOKS

by F. Sokolowski
Montebello, California

Elementary Computer Applications, Ian Barrodale, et. al., John Wiley & Sons, New York, 1971. 254 pages. $8.50.

Not elementary. Contains about fifty programming problems. Contains very detailed discussion of problems. Heavily oriented towards advanced math-science students. Problem topics include roots of equation, numerical integration, system of equations, linear programming, etc. Of little or no use to non-science students. Problems are interesting but require at least calculus background and probably some programming skills not available to many first-course programming students.

Problems for Computer Solution, Steve Rogowski, Creative Computing Press, 39 E. Hanover Ave. Morris Plains, NJ 07950, 1979. Student edition, 100 pages, $4.95. Teacher edition (with answers) 180 pages, $9.95.

Contains ninety programming problems. Gives a fairly detailed description of each problem—one problem per page. Problems are fairly interesting and there is a range of difficulty so both advanced and beginner students can find something. References are provided on some of the problems. Problem topics include: number theory, algebra, geometry, probabilities, calculus, etc. Little for business students, mostly oriented towards math-science type students, but still a nice book.

Computing Problems for FORTRAN Solution, Robert Teague. Canfield Press, 49 East 33rd Street, New York, N.Y., 10016, 1972. 245 pp. $4.95.

Contains 70 programming problems. Problems are suitable for both business and science students. Also the problems could be used by languages other than FORTRAN such as BASIC, PL/1, and maybe COBOL. Most problems do not require much math background. Each problem is described in detail. Many interesting problems.

Program Style, Design, Efficiency, Debugging, and Testing. D. Van Tassel. Prentice-Hall, Inc., Englewood Cliffs, N.J. 07632. 1972. $11.95.

Contains an appendix of 101 programming problems, plus about 100 more scattered throughout the text. Problem descriptions are quite brief but sufficient. Problems are well suited for wide range of languages and range from easy to difficult. Many problems are quite interesting. Contains enough scientific, business, and miscellaneous problems that everyone should find something of interest. Book also covers information mentioned in title. Useful, enjoyable book.

A FORTRAN Program Solver, by Manning, William A. and Robert S. Garnero, McGraw-Hill Book Co., New York, 1970. 167 pp. $7.50.

Contains about 25 programming problems and a short introduction to FORTRAN. A detailed description is given for each problem, and blank pages are provided so one can do his/her work in the book. Thus half the book you buy is blank pages. The problems are very elementary and good only for beginners. Little math required and oriented towards non-science business majors. Personally, I did not like this book.

Programming Exercises, Robert D. Steinbach, Glencoe Press, 8701 Wilshire Blvd., Beverly Hills, California, 1969. 114 pp. $2.95.

Contains about 50 programming problems and a short introduction to computing. Problems are suitable for beginning level programming classes of both science and non-science students. There is a very detailed description of each problem with input data supplied. Problems are grouped according to programming techniques, that is, decisions, iteration, subscripts, subprograms, etc.

Review Editors:
Peter Kugel
John Lees

The Computer and Music. Edited by Harry B. Lincoln. Cornell University Press. 21 articles. xvi + 354 pp.; hard cover. $17.50. 1970.

In a field that is expanding as rapidly as computer-use in music, a book published six or seven years ago may seem hopelessly out-dated. Actually, any interested person reading this book today can gain a great deal of historical and practical knowledge of the field. That is possible because the articles offered by the various contributors are written for various purposes and presume various levels of technical background.

Articles are included that are chiefly philosophical, while others are chiefly descriptive in the technical or research sense. Of the longer articles, there is one that traces the development of technology in its relation to sound and music, and another that surveys computer-composed music to 1970.

Unfortunately, computer-assisted instruction in music is left out entirely. Considering the highly developed state of CAI today, it is interesting that in 1970 the editor of *The Computer and Music* could write: "A third area, computer-assisted instruction in music theory, has emerged too recently to assess its effectiveness or possibilities" (Lincoln, 1970, p. xi).

There is no way that the encapsulated descriptions that follow can do justice to the articles themselves. The point is that, even with the exclusion of CAI, there is in this book a great variety of offerings for musicians, researchers, educators, composers, programmers, and technicians alike. It is hoped that the descriptions will give an impression of the subject with which each contributor is concerned and how the computer relates in each case.

The book is divided into six parts. Part One, *Historical Background,* contains a single twenty-page article by Edmund Bowles on the subject of the development of technology as it relates to music and sound. It is a fine article that uses language with remarkable succinctness. The use of computer-related terms to help describe music-related devices such as the hydraulic organ and the phonograph serves remarkably well.

Part Two, *Music Composition,* contains an article by Brun and an article by Strang that concern themselves with the ethics of computer composition, a fifty-four page article by Hiller that surveys music composed with computers, and an article by Citron that supplies instructions for inputting elements of music to the computer with MUSPEC (a programming language) and describes the processes and results of synthesizing computer output. It is one among several articles that are essentially descriptions of specific projects involving problems of computer input and output.

Part Three, *Analysis of Music*, contains seven articles (each under sixteen pages) all of which are descriptions of specific projects. Perhaps because of the obvious permutational aspects of harmony and the relative ease with which intervals in music can be turned to numbers, many of the projects concern themselves either with searches for the interval combinations in specific pieces of music, identification of style through interval combinations, compilation of statistics related to intervals, or analysis of intervals for the purpose of synthesizing music.

Articles by Fiore and Fuller analyze the music of Webern. Articles by Lefkoff and Stoney each present statistics related in the former case to the discovery of similar segments in the forty-eight permutations of a twelve tone row and in the latter to the problems of equal temperament. The Morton and Lofstedt article describes three Fortran programs that are coupled to numerical definitions of tonal material. Two are for composition and one for analysis of tonal music. They report: "At present, to be sure, both programs produce a conventional four-part music not unlike that achieved by music students at the end of their first year of collegiate study" (p. 161). Roland Jackson's article concludes that the most interesting observation in his study is the selection of certain harmonic colors by Webern, Stravinsky, and Varese which "... are heard frequently enough to provide a sense of unity with the piece as a whole" (p. 146). Youngblood's project, using a modification of the DARMS code, encoded music of Bartok, Schoenberg, and Hindemith for the purpose of establishing composer identification through root progression analysis.

Part Four, *Ethnomusicology,* contains two articles. Benjamin Suchoff reports upon procedures and findings obtained from computer programs that extract comparisons (such as interval sequences) from data encoded with the Ford-Columbia Representation. Pitch, duration, and other elements of music notation are encoded with this encoding system. Music used in the experiments included melodies from the folk song collections of Bartok. The Lieberman article is related to computer recognition of patterns in certain improvisations heard in Javanese gamelan music. The reviewer was impressed by the interesting and practical aspects of this study along with its implications for analyzing improvisatory styles and its slight overtone of what might be called an ethical problem. There is a thread of concern, in this book, about relinquishing decisions to the computers. As might be expected, the composer-contributors seem to address themselves to this concern more than the researcher-contributors.

Another thread that winds through the book is the search for some further definition of the concept of "style." Depending upon how well elements of style are defined, the computer ought to be able to identify composers and periods given the encoded musical elements of a piece. It ought even be able to output data that can be synthesized to produce pieces in various styles.

Part Five, *Music History and Style Analysis* contains five articles. Two of the articles are directly concerned with using the computer to further define elements of style. Crane and Fiehler report upon numerical or statistical methods of comparing musical styles. They are able to picture some of the results of their work with a dendrogram showing twenty chansons clustered according to style. "Music Style Analysis by Computer" by A. James Gabura is one of the longer articles (53 pages). Gabura is able to describe clearly the use of such concepts as "key" based on "pitch-class distribution" and "root movement," which are concepts experimented with for the purpose of developing computer-based style analysis. Gabura also describes methods for extracting such concepts or "parameters" from pitch and duration data for the purpose of identifying individual style in the pianoforte music of Haydn, Mozart, and Beethoven.

The remaining three articles constitute practical projects in the area of Music History. Barton Hudson describes a proposed catalog of French chansons inputted with the DARMS encoding system and accessible by musical incipits and other references. Earle Hultberg reports on programs that transcribe tablature to standard notation. Theodore Karp reports upon a computer-based system for determining degrees of melodic resemblances in Notre Dame *organa dupla.*

Part Six, *Music Information Retrieval,* contains two articles. The first (26 pages) by Michael Kassler is taken up chiefly with "... a description, in the form of a programmers' manual, of a special-purpose programming language called MIR — the acronym of the phrase 'musical information retrieval'" (p. 299). The second is a detailed proposal for the development of a computer-accessible music library catalog for scores and phonorecords. The authors (Tanno, Lynn, and Roberson) are not referring to retrieval of individual notes in compositions, of course, but to literal library references such as "composer," "instrument," "subject." They make a strong case for such a system claiming not only increased efficiency, speed, and accuracy, but lower cost. Most important to them, however, are "... the myriad of information possibilities inherent in the data base being developed" (p. 342).

It is certainly outside the scope of this review to update the projects in this book. It is hoped that a new and updated edition of *The Computer and Music* will soon be forthcoming.

Robert W. Placek
Assistant Professor of Music
University of Georgia
Athens, Georgia

* * *

The Technology of Computer Music. Max V. Mathews. M.I.T. Pres, Cambridge, Mass. 188 pp. 1969

The term "computer music" has been used to describe computer-aided activity in all phases of music production. The subject of Mathews' book is not that broad. It is concerned primarily with a specific method of computer sound synthesis, the direct calculation of the sound pressure wave. I have come to refer to this method as *direct synthesis.* Other methods exist which involve a computer in control of external sound producing and sound shaping modules. These methods might be referred to collectively as *indirect synthesis.* The main advantage of direct over indirect synthesis is that, at a greater cost in CPU time, the composer is freed from the limitations imposed by the availability and configuration of the external sound modules. With indirect synthesis the number of each kind of module is finite. By dealing directly with the sound wave the composer has virtually an infinite supply of computer-simulated modules, giving him virtually infinite control of the sound. He can produce any sound he can specify and vary it over time in any way he can imagine. The trick is in being able to specify what is desired. The MUSIC V program described in this book, a product of the work of Mathews and others at Bell Labs, provides the composer with a powerful language for doing just this.

Chapter One, "Fundamentals," describes the technique for representing sound as a sequence of numbers, known as samples, and then transforming the numbers into sound. Each sample represents the amplitude of the sound wave for an instant of time. Once the samples have been generated, they can be transformed into a variable voltage which can then be amplified and used to drive a loudspeaker. These discrete samples represent a continuous function varying over time in the same way that the individual frames of movie film represent a continuously changing picture. The speed at which the individual units are presented is so fast that the observer cannot resolve them; he is aware only of continuity. Various kinds of distortion arise if the range of numbers used to represent the amplitudes is not wide enough (quantizing error) or if the duration of the sample is not short enough (sampling error). 30,000 samples per second must be calculated if the full 15,000 herz bandwidth of human hearing is to be represented. Attempting to generate sounds with partials higher than half the sampling rate results in another kind of distortion (foldover).

Chapter Two consists of a graded sequence of scores written in the MUSIC V language. Through these examples Mathews describes the operation of the program and gradually introduces various components of the language. MUSIC V is written for the composer accustomed to designing his instruments by patching together the sound modules found in conventional synthesizers. In MUSIC V the patching is done symbolically. Information for playing the instruments is given by the *notes,* data records of pitch and time specifications. Although the input format is tedious, the composer can write conversion and composition subroutines in FORTRAN or assembly language which allow the computer to take over much of the "dirty work" while the composer works with a more musically-oriented format.

Chapter Three, "MUSIC V Manual," presents much of the same information as Chapter Two but in a systematic manner, designed for easy reference.

Although the book is now seven years old and MUSIC V has undergone further modification since its publication, it still has much to offer. There are similar programs running on institutional computers all across the country. This book is a good introduction to any of them. Chapter One is a thorough guide to the sampling technique. It and the Appendix contain the mathematics required for a thorough understanding, yet the chapter is comprehensible to those who have not progressed beyond a basic understanding of algebra. The mathematical sections have been flagged with asterisks, and the rest of the chapter has been so arranged that the reader can skip the math and come away with a working knowledge. Mr. Mathews has accomplished this feat by clear and non-technical explanations supplemented with copious illustrations.

The book could be used as a text. There are lists of sample problems and bibliographies which lead the reader to some of the most significant literature on the subject wirtten before 1969. Since it is almost entirely in FORTRAN, MUSIC V is easy to implement on any system and also easy to study. Consequently, the book and the program together provide a model for anyone intending to write a sound synthesis program.

The criticism offered here is not of the book itself but of the kind of program which has been described. Direct synthesis programs are extremely powerful, but they also require large amounts of CPU time. It is quite conceivable that a given second of sound may take a full minute to compute. If the composer has access to a computer for the time required, he then must contend with the special requirements for converting the digital representation of the sound to a form that can be recorded and played back by a conventional tape recorder. Sometimes these facilities are not available at the same computer center which produced the digital tape. This can cause long delays between submittal of a job and the return of an audible product. However, advances in computer technology will probably deal with these problems before a more powerful sound generation technique is devised. In Mathews' book this technique is given a classic presentation.

Richard E. Saalfeld
Columbus, Ohio

* * *

Rogers, John E. "The Uses of Digital Computers in Electronic Music Generation," from *The Development and Practice of Electronic Music* Edited by Jon H. Appleton and Ronald C. Perera. Prentice-Hall, Inc., Englewood Cliffs, N.J., 1975.

It is exciting to find a book on electronic music destined for wide acceptance which devotes almost one-third of its content to computer use for electronic music generation. Mr. Rogers places a deserved emphasis on the computer when he suggests that the "expansion (of electronic music studios) must involve digital computers as essential units in electronic music generation." This support of computer use might even be considered mild, since it is the reviewer's belief that, even more than as an expansion, the computer will be *the* central and basic unit in future electronic music studios, ultimately requiring their total re-design.

Rogers cites two basic methods for the computer's use: as a digital control device for an analog studio and for pure computer sound synthesis. The logical trend to systems based on digital control of digital devices is referenced, but both enthusiasm and material on this approach are lacking in this chapter of the book.

Giving an excellent recommendation to this work is easy, since it provides valuable, specialized information not frequently found in a book with such wide circulation. However, the reader must be able to move from the interpretation of general information usually understood without in-depth knowledge to the fairly detailed descriptions appealing mostly to those who have a specialized interest in computer music. The transition from general to specific is quick and apparent, with enough material included to develop each into separate and valuable chapters.

The general information covers: the limitations of present-day studios which may be eliminated through computer use; the basic uses of computers in electronic music; the major characteristics of computer technology such as timesharing, batch processing, and minicomputer applications; and computer programming techniques. In addition, a replication of information readily available and clearly presented in Max Mathews' *The Technology of Computer Music* (M.I.T. Press, 1969) is included. This fact is mentioned, not to criticize Mr. Rogers for including his own explanations of D to A conversion, foldover distortion, sampling, quantizing, and other basics inherent in digital representation of sound waves, but to emphasize the importance of Mathews' earlier work now recognized as a published landmark in the field.

Several explanatory points related to simple programming concepts were disappointing when compared to many other excellent sections of the chapter. For instance, the technique of branching, with all of its unique and valuable characteristics, was presented too simply as a mode which "allows certain instructions to be skipped or branched around." In a somewhat similar fashion, the notion that "the computer should ... be programmed in a high-level language, preferably Assembly Language" is puzzling in that much more sophisticated languages are available, even on minicomputers, for executing a series of assembly instructions in one single command. Rogers' justification for use of assembly language due to economic factors associated with its greater speed seems invalid with modern technology.

The more technical aspects of the chapter dealt with computer sound synthesis, based (too much) on the MUSIC 360 coding system of Barry Vercoe. Although the reviewer considers this as a weakness of the article, Rogers purposely limits himself and states, "MUSIC360 can be understood, at a basic level, with only the information presented in this chapter." The weakness here that is felt is one of narrow coverage of "The Uses of Digital Computers in Electronic Music Generation," since other different approaches have equal validity.

Although I would have preferred it if Mr. Rogers had looked more toward the future — to advancements that are bringing computer technology out of its infant stage, to general availability for all, and to the increased ease of use and operation, he has made a major contribution in reporting some of the continuing developments in the field.

David Swanzy
Professor and Coordinator of Graduate Studies
in Music Education
Southern Methodist University
Dallas, Texas

☆ ☆ ☆ ☆ ☆ ☆ ☆ ☆ ☆ ☆ ☆ ☆

Computer Careers, Planning, Prerequisites, Potential. John Maniotes and James S. Quasney. Hayden Company, Inc., Rochelle Park, New Jersey 07024, 180 pp. $4.95. 1974.

The eight chapters of this book provide the reader with a sound basis for making decisions concerning a career in electronic data processing. The second chapter, titled "How Computers Do It," provides an excellent overview of computer operation written in plain language which can be understood by individuals without a background in electronic data processing.

Other chapters cover such areas as the kinds of educational training required for the variety of jobs in EDP and do an admirable job of outlining the requirements of various programs. Unfortunately, the book does not contain information on educational computing, which is rapidly expanding as a "new" enterprise. Chapter 5 provides an excellent overview of the costs of an education and how the student can defray such costs through grants, scholarships and fellowship programs. The resource list provided is extensive in terms of the agencies which assist students enrolled in EDP fields.

Chapter 6 discusses the problem of finding a computer-EDP-oriented job and presents an excellent outline of a procedure for developing a resume which can be used universally in preparing such a document. It also provides excellent hints on the interview process and the variety of tests that one may be required to take in seeking a position.

The final chapter of the book describes three types of institutions that offer degrees in electronic data processing. It gives the pros and cons of each type of institution and points to the need to look critically at all institutions to determine if the objectives of the program are in harmony with the career objectives of the individual.

Although texts typically involved in computers and computer professions are out of date within a few years of publication, this text can provide a continuing source of information for students who are graduating from high school and trying to determine which institution they should attend, as well as students who are already in higher education who are looking forward to a position in the field. The book may seem elementary to many, but this reviewer feels that it does provide an excellent resource for a younger student who is in the awesome position of trying to make a decision for a life-long career. I would also recommend the book for guidance counselors, since it is written in terms that non-EDP-oriented people can understand and apply.

Daniel Krautheim
Columbus, Ohio

☆ ☆ ☆ ☆ ☆ ☆ ☆ ☆ ☆ ☆ ☆ ☆

Introduction to Programming Languages. W. Wesley Peterson. Prentice-Hall Inc., Englewood Cliffs, N.J. 358 pp., $12.95, 1974.

In this easy to understand book, eight programming languages are presented under the four major sections *Scientific, Data Processing, Character String Processing,* and *List Processing.* The languages used are BASIC, FORTRAN, COBOL, PL/1, ALGOL, APL, SNOBOL, and LISP. PL/1 is covered in each of the four major sections in which the language features appropriate to that section are discussed. The use of simple problems to illustrate the features of the language under discussion makes this an attractive text for an introductory course on programming languages. Some of the more advanced techniques such as recursion and list processing are clearly explained. An additional aid is the use of the same set of problems for each chapter (on language) within a major section. Some will complain about the lack of exercises for each chapter, but with the various illustrations, alternative problems should easily come to mind.

The book is a good reference work, but it is IBM orientated, and the implementation for a language will vary from one vendor to the next. Therefore one must consult the system's reference manual before using the language. I found the author's coverage of PL/1 to be more than adequate and his section on PL/1 use of based variables in list processing to be outstanding. I recommend the book as a text in a survey course on programming languages or as a reference work for the practicing programmer.

William J. Marshall
Chelmsford, Mass.

The Best of Creative Computing, Volume 1, edited by David H. Ahl, 317 pages, $8.95 Creative Computing Press, P.O. Box 789-M, Morristown, NJ 07960. 1976.

In the preface, the editor states the material in the book is *"diverse."* That comment is not an understatement. There is material and information for everyone interested in computers. Just as the material originated from a wide range of sources, so its appeal is also far-reaching. The table of contents indicates the breadth of material from articles and commentary to fiction and poetry, a little foolishness, puzzles and problems as well as a wide array of computer games and book reviews.

The book consists of the material first seen in Volume 1, issues 1 through 6, of *Creative Computing.* The material has been collected under the headings indicated above, but retains the exciting reading always present in each issue of *Creative Computing.*

The book contains something for people of all ages, from games for the young to puzzles and articles designed to keep one in deep thought for hours.

The book is recommended for all ages as a volume that can be picked up and read anywhere for any length of time. For those who have been late in subscribing to *Creative Computing,* this volume is now the only source of those back issues of Volume 1 that have been sold out.

John J. Jackobs
Coe College
Cedar Rapids, Iowa

* * * * *

BASIC Software Library, Volume II Engineering and Statistics. R.W. Brown, 260 pp. paper, $24.95. Scientific Research Instruments, PO Box 2096, Ashland VA 23005.

The BASIC Software Library Volume II is a compilation (if you'll excuse the term) of forty-four applications programs in the areas of engineering and statistics. An ad for this series of books in a hobby computer magazine tells us, "The intention of this work is to allow the average individual to easily perform useful and productive tasks with a computer... This library is destined to become one of the reference bibles for the small computer field...." The ad further states that ALL the programs are written in compatible BASIC executable in 4K MITS, SPHERE, IMS, SWTPC, PDP, etc. BASIC compilers. All right, so everyone gets a little carried away sometimes. But since many personal computer owners *are* looking for applications software, how good is this book?

The programs themselves range widely in degree of generality. Almost anyone can use a program which graphs two functions on a TTY. However it seems quite unlikely that the average personal computer user is going to need to calculate the thickness of a steel beam. The *BASIC Software Library* does not attempt to explain the applications it covers—nor should it—since you should already understand your application before you try to apply a computer to it. If you're into mathematics or statistics, or need to plot functions or data, this book may be for you. If you have a very specific application you'd probably better have a look at the table of contents before you buy.

As far as the programming techniques used, most of the programs seem reasonably efficient and we especially liked one that produced random numbers without RND, in just 12 short lines of BASIC. One program which caculates positions of stars was annoyingly inefficient using the following technique for printing the name of star #W.

```
IF W=1 THEN AAAA
IF W=2 THEN BBBB
IF W=3 THEN CCCC etc.
  and then to print the names:
AAAA PRINT "name 1"
     GOTO XXXX
BBBB PRINT "name2"
     GOTO XXXX
CCCC PRINT "name3"
     GOTO XXXX
```

and so on, which could be done more efficiently with DATA statements and a read loop. Additionally the programs are not nearly as transportable as stated, since they sometimes use user-defined functions, exponentiation, character strings, and mass-storage files. There's nothing wrong with that, but you shouldn't say that *all* your programs run in 4K BASIC on a micro if they don't.

$24.95 is a lot of money to pay for a softcover book, especially when some of the pages are almost too light to read, and in other places corrections to the original program run have been written in by hand. If you need to compute integrals or linear regressions or even thicknesses of steel beams, then the *Software Library* could save you a lot of time spent in research, program writing, and debugging. But before you buy it, you should seriously consider whether or not the book will help you with your application.

Steve North
Newfoundland, NJ

* * * * *

CMOS Cookbook, by Don Lancaster. Howard W. Sams & Co., Inc., Indianapolis, IN 46268. 414 pp., paperback, $9.95. 1977.

If your logic projects have been thwarted because you didn't have the regulated power supply that was required, or you were afraid that you'd goof up the assembly and blow up the whole circuit, or if you're fed up with all the hassles of TTL, then CMOS is for you, and Don Lancaster's *CMOS Cookbook* is for you.

CMOS is a very easy logic family to use. It can even be fun for once to build all those logic projects. You don't need to worry about the power supply; CMOS tolerates unregulated, noisy power supplies and helps out by using very little current. Even with all its nice features, CMOS is very low in cost.

This book covers the typical applications for a logic family and some features which make CMOS particularly useful in certain applications. The uses for CMOS that have no counterparts in any other logic family are detailed, as are its few disadvantages.

The first chapter supplies all the background information on CMOS — how and why it works, some unique features, usage rules, power-supply design, and general practices to be followed in design. Chapter Two is a hundred pages of individual descriptions of CMOS devices. Each description gives you only the information you need to know to use the device, and the problems and restrictions in using any device are clearly laid out so that you won't have any surprises later on.

Chapter Three covers logic design starting with the basics and moving up to the use of transmission gates and tri-state buffers. The latest techniques of redundant logic design using data selectors, ROMs, PLAs, and microprocessors are presented as approaches to simplifying complex problems that previously could have required high cost and much time spent on design and modification. It is in this chapter that Mickey Mouse logic is introduced as a trick for simplifying some designs.

Chapter Four describes CMOS multivibrators. Five develops clocked logic from fundamentals and describes the available CMOS flipflops and their applications. Six carries the applications to larger-scale sequential design using shift registers and counters, detailing use of these in computer and music applications. The use of CMOS in some unique ways is the subject of Chapter Seven, where you learn of CMOS used in operational amplifiers, bidirectional analog switches and phase lock loops.

The last chapter is devoted to full-scale applications of CMOS. These include timing and timekeeping circuits, a frequency counter, a video-game circuit, music circuits, computer circuits, and some challenges for you, the CMOS logic designer. An appendix with suppliers and addresses is included.

You'll find that this book is one of the most useful and informative books you can get. After reading it, you'll not be able to wait until you can start working with CMOS.

Dennis Keats
Hopkins, MO

* * * * *

Model Railroad Electronics, by James Kyle, TAB Books, Blue Ridge Summit, Penn. 17214. 307 pp., paperback $5.95. 1977.

Covers most aspects of model RR electronics, including the use of digital logic.

* * * * *

Programming Proverbs For Fortran Programmers, Henry F. Ledgard. Hayden Book Company, Inc., 50 Essex St., Rochelle Park, NJ 07662. 125 pp. $5.95. 1975.

Programming Proverbs provides the basis for a second step in the development of programming skills. It is for persons who have some familiarity with Fortran, but do not yet have a satisfying level of proficiency. The book is intended to be a guide to better programming, not an introduction. The author's intent is to promote improvement through the observance of 25 essentially simple rules called *proverbs.* Presentation of each rule is accompanied by recognition of a condition found in the work of an inexperienced programmer. These are followed by a discussion and examples showing application of the rule. In most instances these rules are clearly worth consideration by fledgling Fortranners. Although he fashions his message around a group of rules, the author carefully notes that he does not intend to remove opportunities for creativity in programming. He introduces his rules with a *prefactory proverb:* "Do Not Break the Rules Before Learning Them." This suggests the spirit prompting the book.

While showing readers how to improve their Fortran programs, the author gently raises the sights of Fortran-only programmers leading them toward an appreciation of other programming languages. In two series he presents equivalent programs in Fortran, BASIC, PL/1, and SNOBOL or COBOL. This is done in a way that is easily grasped by a Fortranner, yet leaves an unmistakable impression that other languages do indeed have advantages in certain situations.

This reviewer found the author's explanation of the context effects of function subprograms to be especially enlightening. This feature alone would be worth the cost of the book and the time to peruse *Programming Proverbs.*

Thomas A. Boyle
Purdue University

(Available from the Creative Computing Book Service. See coupon elsewhere in the magazine.)

* * * * *

Computer Graphics: 118 Computer - Generated Designs. Melvin L. Prueitt. Dover Publications, Inc., 180 Varick Street, New York, NY 10014. 69 pp. $3.00. 1975.

A three-dimensional U.S. flag rippling in the breeze, soaring spires, and ethereal multicolor creations floating in black space are just a few of the stimulating computer-generated artworks that await the reader of this delightful book.

Following a short, thought-provoking discussion of computer art, the author moves directly to examples created by PICTURE, a program of his own invention. The pictures are accompanied by brief, informative comments that cover such topics as hidden-line removal, perspective, and optical illusions. A number of the pictures show the three-dimensional representations of explicit mathematical functions while others depict unpredictable patterns resulting from letting the computer generate random data for output. Unexpected images caused by errors, both human and mechanical, contrast with the precision of three-dimensional plots of magnetic fields and nuclear spectra.

Anyone with an interest in the potential of computers aiding human thought and creativity, especially in the area of computer-generated art, should get their hands on this excellent book.

Jay Wooten
Westford, MA

* * * * *

Graze Ecology Simulation. Michael Chester. Hewlett Packard Computer Curriculum, 1501 Page Mill Road, Palo Alto, CA 94304. Student Text, 23 pp.; Teachers' Notes, 29 pp.

The student is given a range, 3 square miles in area, which has 200 rodents and 10,000 grasshoppers per acre. The student's objective in this computer simulation is to raise cattle on the range while maintaining a balanced ecology. The student inputs the number of cattle to be raised, and also the number of songbirds and hawks he or she thinks is required to maintain the balance. The populations of all species are then computed in half-year increments over a 15 year period and printed out, along with a score from 0 to 100 which measures the student's success in achieving the objective. By analyzing accumulated information, the student can then try to improve his or her score by systematic variations of initial species populations.

The author provides a good qualitative description of the model, along with suggestions for additional projects and several references for further reading. The value of the simulation to the more serious student would be enhanced by a discussion of the equations used, since many will want to tinker with the model. (For example, when grasshopper populations "explode," the computation terminates with an "out of control" message and a zero score for the user. A problem solving student may be more interested in knowing what the grasshopper limits would be in the absence of predators than improving his or her score.)

The simulation can be studied at many levels and it can therefore be recommended for classroom use from junior high through college. The BASIC program with external documentation is provided.

Scott Davidson
Silver City, NM

* * * * *

Hobby Computers Are Here!, Edited by Wayne Green, 96 pp. paper, $4.95. *73* Magazine, Peterborough, NH 03458.

Since amateur computing is expanding so rapidly, there are a great number of people who want to know how to get their own terminals and computers up and running. *Hobby Computers Are Here!,* a "book" which has fewer pages than an issue of *Creative,* purports to fill this need. On the opening page Editor Wayne Green has the audacity to declare, "This book is one of the few (if there are any other) sources of information on all aspects of computers ... hardware (with very simple explanations of the basic circuits involved) ... the software ... *and* systems." Well, we'll all have to throw out our other computer textbooks.

Having recovered from the shock of reading that, one quickly discovers that this book consists of nothing but reprints from *73*, a ham radio magazine. Some of the articles which explain the fundamentals of digital electronics are quite helpful. Unfortunately, construction articles such as "A Morse to RTTY Converter" are definitely not of interest to the mere amateur, non-ham computer owner. The emphasis on ham radio in this book might be attributed to the fact that almost all the articles were written by hams. *Hobby Computers Are Here!* also contains twenty-one editorials by Wayne Green. They're generally centered on amateur and ham computing but occasionally Mr. Green rambles off on some tangent. Some readers may enjoy this style but some won't.

One of the problems with taking articles and editorials from a magazine and then printing them in a book is that it's not always apparent to the reader what's current fact and what's dead history. For instance, what is a novice going to believe when he reads, "None of the present-day tape storage systems are ideal for small computers and the race is on to invent a mass memory storage system which is geared to the low cost computer." Maybe he'll keep on reading and somewhere later in the book discover that cheap tape systems do exist. What if he doesn't? It's up to the reader to sort out the here-and-now from obsolete information.

Green's admission that "There are fantastic opportunities in the small computer market for making large gobs of money," seems just a bit too frank since there are eight pages of advertising in the middle of this "book," not to mention another ad on the back cover. At $4.95 *Hobby Computers Are Here!* really tells us, *Overpriced Hobby Computer Books Are Here!*

Steve North
Newfoundland, N.J.

* * * * *

Computers and You. Kurt R. Stehling. Mentor, paper. 1973.

One of the many books for popular consumption to provide an understanding of the use and effects of computers on the whole range of human endeavor. Better than some, but disappointing in the amount of language one so regularly encounters in departmental reports. Well selected topics, informative treatment, but it still comes out fact stacked on fact stacked on fact and is not likely to generate interest in naive readers, though it does provide the type of introductional material so needed by the public.

John L. Randall
Kensington, MD.

Computing with Mini-Computers. Fred Gruenberger and David Babcock. Melville Publishing Company, Los Angeles, California, 288 pp. $13.75. 1973.

The authors' effort to provide an introduction and overview in the first few chapters of *Computing with Mini-Computers* makes an attempt to define a mini-computer based on size and cost. However, in the three years since the publication of this text, there have been many changes in the mini field. A mini-computer is defined in the text in terms of three variables — storage capacity, top speed, and operation-code repertoire. In spite of the fact that physical parameters are the least important, the authors go on to describe the mini in terms of its physical size. However, mini-computers have been configured in a variety of ways which greatly exceed the physical dimensions provided in the book. The examples utilized throughout the text are for a Varian machine, even though it is pointed out that Digital Equipment Corporation both produces and sells the largest number of mini-computers. There are some commonalities between mini-computers. However, the differences exceed the parameters defined in this text.

The chapters on flow-charting are relatively standard and can be found in other texts. The book uses a problem-solving approach so that the reader is led through a series of problems that require computer solutions. Lack of access to a mini-computer might be a limiting factor for many potential readers who are interested in the field of minis but do not have access to such equipment. Although the book is well developed, it discusses material that can be found in many other texts. It is the opinion of this reviewer that the textbook has only limited utility for individuals interested in mini-computers since the state of the art has changed drastically since 1973. There is virtually no discussion of micro-programming units which may or may not be classified as mini-computers. The basic and historical information contained in the book is of some value and the conceptual information on indexing, sub-routining and sequencing of program statements can be applied to a variety of situations. The chapter on testing and de-bugging of programs is relatively standard. The eight-page glossary of computer terminology is fairly complete for 1973. However, it lacks comprehensiveness for modern mini-computing. I would recommend the book with reservation for individuals who are interested in mini-computers as a fair resource and entree to some of the more current literature found in the periodicals.

Daniel Krautheim
Columbus, Ohio

* * *

Finite State Fantasies. Rich Didday. Matrix Publishers, 207 Kenyon Road, Champaign, Illinois 61820. 50pp., $2.25, 1976.

Finite State Fantasies is a comic book devoted to visual communication of computer user situations. About half the booklet explains how the computers work. This part is good for anyone who would like to develop an understanding of computer hardware.

The other half of the booklet involves pictorial representation of common computer user occurences: bugs, glitches, and computer obsession. Interspersed in the booklet are several one-page cartoon series.

The booklet can be understood by someone of any age group or computer background. The graphics are nicely done. The book is a nice addition to the field of fun computer books and it would make a nice gift.

Dennie Van Tassel
Santa Cruz, California

Background Math For A Computer World. Ruth Ashley. John Wiley & Sons, Inc., 605 Third Ave., New York, NY 10016. 286 pp., paperback. $3.95. 1973.

This book is one in a series of Wiley Self-Teaching Guides. The format of the book is programmed instruction. Each chapter is divided into individual frames of instruction with their own questions to check on the reader's comprehension of the text material presented. Each chapter concludes with a self test on the contents of the chapter. The book concludes with a final test on all the material in the book. Answers to all questions are provided in the text.

The author intends the book "for the tens of thousands of people who find that their lives are being increasingly affected by computers. It is for the students with no college math and very limited high school mathematics who discover that they, too, are expected to be able to use computers — in business, in psychology, in education, in the social sciences."

The first two chapters concern themselves with the binary, octal, and hexadecimal number system. Operations within, and conversion between these systems are covered. I would not think that these topics are important to the type of person who would be using this text. The third chapter concerns itself with logic for computers. Conjunctions, disjunctions, negations, implications, De Morgan's Rules, and truth tables are some of the topics considered. The fourth chapter deals with being able to follow, but not write, a flowchart. Chapter 5 covers integer arithmetic, floating-point arithmetic, scientific notation (E-notation), and operations in E-notation. The last seven chapters are separated into the following seven topics: Interest and Mortgage Problems, Sequence and Series, Probability, Statistics, Linear Equations, Matrix Algebra, and Game Theory. Some of these later chapters, such as the chapter on Statistics, do not explain concepts well, and so it becomes a matter of accepting certain formulas on faith. I feel that the chapters on Logic and Interest and Mortgage Problems are the best in the book.

It is hard to think of this book being suitable for "tens of thousands of people." It may be helpful to some college students majoring in the humanities who find themselves exposed to the computer, but only a subset of the chapters would prove helpful. Another possible use for the book would be as a supplement to a programming book in an adult education or equivalent course.

Bruce W. De Young
Oakland, New Jersey

* * *

FORTRAN Techniques. A. Colin Day, Cambridge University Press, New York. 96 pp. $3.95, 1972.

This book's subtitle is "special reference to non-numerical applications" for good reason. In a slim 96 pages, the author covers many salient points on lineprinter graphs, table searching, keyword-in-context identification, stacks and queues, list processing, and sorting. The book fills in knowledge on a set of random techniques used by FORTRAN programmers in each of these areas; it is a practical book rather than a theoretical one.

The first chapter contains descriptions of basic techniques such as flags and switches, packing and unpacking numbers, table translation, buffers, open-coded subroutines, and simple character manipulation hints which will work on even the most restrictive FORTRAN compliers. The chapter on line-printer plotting includes remarks on point, line, density, and histogram plots, the chapter on sorting includes four sorts, and the chapter on stacks and queues includes a description of simulating recursion in a FORTRAN program. The only fault I can find with the book is that Day wastes a chapter on "symbol state tables" (useful to check that data conforms to a set of syntactic rules), giving it too abstract a coverage for such a book. All in all, Day manages to pack copious hints and notes (including *examples* and diagrams) quite well into a slim but concise reference work.

Brian N. Hess
Western Springs, IL

Reflections

Reflections At the End of Our Third Year

From time to time I get questions from people at conventions, and from readers through the mail, as well as from my own staff, as to what the philosophy of *Creative Computing* is. Where are we going? What's the magazine trying to be? And as we enter our fourth year, I suppose it's an approporate time for the publisher to sit back and speculate a little bit on where this thing is going, this monster I created about four years ago, that in some ways has gotten completely out of hand.

Our direction over the first three years has obviously changed. This is probably most apparent when you consider two major groups of readers, educators and home-computer users. Some educators have complained that we have given up the education market and we have become hobbyist-oriented. To this I would say, nicely but bluntly, that (1) you educators are rather narrow-minded if you think that the types of things *Creative Computing* is running are not appropriate for education just because they happen to focus increasingly on microcomputers and on manufacturers who don't have the vast dealer networks or army of salesmen to be calling on schools. The educational value of a computer may be far greater if the student is dealing with an accessible, hands-on micro than if he's dealing with a computer that can't be touched by anyone but a "qualified field-service technician."

And (2) I would ask the skeptical educator: aren't you happy to have other people reading a thoughtful and pedagogically sound magazine? Maybe *Creative* doesn't look like a typical educational magazine anymore because we're not running the dry scholarly articles that "should" appear in an educational magazine. However, we are running ideas that have a sound basis but we're presenting them in easy-to-understand terms. This is rarely found in educational magazines which somehow feel that to put things in four-syllable words makes them more acceptable to faculty members who probably don't have more understanding of these fancy words than the student, and certainly aren't any more interested or inclined to read them.

Actually we have very, very few educators who are criticizing *Creative Computing* and I hail the foresight of the majority of our educational readers.

But that doesn't get at the question of where we're going and what our philosophical basis is. We have many diverse groups of readers: educational computer users, hobbyist users, time-sharing users, users of minis or micros, and big-system users. I guess the one common denominator in all of those descriptions is the word *user.* The assumption that we make at *Creative Computing* is that our readers are interesting in *using* their computer power. They are not tinkerers. They are not electronic nuts. Oh sure, they may well tinker around to get their system working. They may well be immersed into electronics, but the main object of their purchasing a computer in the first place is not to solder boards together forever, or to learn about various communications protocols, or to build a fancy woodgrain cabinet for their finished machine, but it is to use it for some application. Be that personal, or educational, or for home-management, or building management, or for small business. The point is that the readers of *Creative Computing,* we believe, are interested in applications.

Now, there are two levels to the applications. Some applications are original and unique enough to a particular reader that he or she is the only person that can or will produce it. For these readers, we feel that our obligation is to provide programming techniques to make writing their application easier and more efficient. Techniques like sort routines, large-number processing, recursion, shuffling routines, and data-based management schemes. Also for the reader who is writing his own application, we present materials on different computer languages. Perhaps one application could be better implemented in APL, or Fortran, or perhaps the version of BASIC that is available to the reader is not compatible with that particular application because it doesn't have the proper string-handling or file-handling capability. Consequently, we feel that another one of our jobs is to take a look at different types of BASIC compilers and their strengths and weaknesses. Ditto other high-level languages.

In addition, for the reader who is writing his own application, we feel that it's worthwhile to present articles like "Computational Unsolvability," or "Catastrophe Theory." The first discussed certain types of problems which are not amenable to computational solution. The second looked at some completely unexpected types of curves that provide a framework for analyzing discontinuous events in fields such as biology, sociology or even the stock market.

For another group of readers who are not so much interested in writing their own unique application, but rather in using finished applications or perhaps combining several to do something of use, we present a certain number of complete programs. So far, most of these have been in the area of games, puzzle solving, and various other recreational applications. In the future, we expect these applications to broaden out into more serious areas such as text editing, word processing, building management, household management, and business functions. Of course, we'll continue to present games and recreational applications since these have proved extremely popular in the past and, by my best estimate, at least 70% of the home users are using their computers almost exclusively for games and leisure.

Along with applications, I think it's equally important for one to consider what is really worthwhile doing with a computer. There are lots of things that *can* be done, but what is really *worth* doing? Now there is no hard-and-fast, cut-and-dried answer to that question. Every person is going to have to decide on his or her own part what is worth doing with his (or her) computer. However, from time to time, we will present articles and even fiction and poetry which will speculate on future computer applications and where they might lead. Of course, we would hope that any fiction or poetry that we would put in the magazine would be interesting and entertaining in its own right, but if we choose our material carefully, maybe we can also provide a message in this type of material.

Since we feel that our readers have a user orientation, one other mission that we feel is appropriate is to present accurate reviews of books and other materials aimed at a user. As a result, we are the only publication going into the hobbyist market that has any kind of serious book-review section. If you read these reviews, you'll realize that many of them go far beyond reviewing the specific book in question, but present ideas, commentary, and thoughts of the reviewer. Our reviews, in other words, are considerably closer in character to those appearing in the *New York Times Book Review* than to the *ACM Computing Reviews.*

I think it's interesting to note that our overlap in readership with other magazines is heaviest with *Scientific American* by a rather wide margin and then, secondly, with *Byte* magazine. What this indicates to me is that we have a very well-educated reader and one with a rather broad set of interests. Of course, they are interested in the hardware; many of them who are not in an environment with ready-made computers and terminals at their disposal are trying to construct their own systems and consequently have the need of a hardware-oriented magazine such as *Byte.* However, I think they realize that eventually their system is going to get up and running, or perhaps they *hope* it is, and at that point, then they're going to need an applications-oriented magazine like *Creative Computing.*

Some months ago, we had a major debate among our staff members and advertising reps whether or not *Creative Computing* should become a monthly magazine. There were good arguments on both sides. However, what it finally boiled down to was the fact that we were making a small amount of money and we had to choose how to invest that money most wisely. If we had become a monthly magazine, we would have had to put considerably more resources into soliciting advertising, since very few magazines can survive on subscription revenue alone. In other words, all of our excess capital for at least the next nine to twelve months would have had to go into catering to, and soliciting, advertising. Alternatively, we could put the same money into improving the editorial quality and hiring additional people on the editorial side of the magazine. This is what I elected to do. *Creative Computing* now has a larger editorial staff of full-time writers and editors than any other educational or hobbyist computer magazine, bar none. For the foreseeable future, therefore, my intent is to improve upon the editorial side of *Creative Computing.* My intent is for it to be of the absolute highest quality. That is not to say there will not be occasional typos, or errors in program listings or other gliches. However, we have elected to make *Creative Computing* a leader as a result of its editorial quality rather than its advertising content. My belief is that, eventually advertisers will discover that *Creative Computing* readers like the magazine for its editorial quality and therefore will advertise with us and one will follow the other. On the other hand, its a risky business going out and soliciting advertising, hoping that you get it, and then when you do, hiring the editorial people to beef up the quality of the magazine. One reader termed us the *Scientific American* of the personal computing field. That's a very complimentary, but I think very apt, description of *Creative Computing.*

Where do we go from here? Well, certainly more applications in a broader set of areas, more programming techniques, more speculation on what should be done with computing power, more reviews, and, oh sure, some more fiction and foolishness too, because along with all of the pedagogically sound serious material, there is a streak of foolishness in all of us. Or if not in all of us, there at least is in me, and as long as I'm the publisher, you're going to continue to get some cartoons and triviality along with the serious stuff.

But we'll be evolving too. We'll be evolving in response to technological advances and also in response to changing desires of our readership. The only way we know about these changing desires is from your cards and letters, so keep them coming. Oh, we also know from subscription cancellations and I'd just as soon you *not* keep them coming. Actually, we have an exceptionally high renewal rate compared to the magazine industry at large, which pleases me greatly. However, we're not perfect. I recognize that and I'm happy for any comments that will bring us a little bit closer to what you, the readers, want to see in *Creative Computing.*

It goes without saying that articles along the lines mentioned above, or dealing with new things that you think would be suitable for *Creative Computing,* are always welcome. We pay for articles; you'll never get rich and you won't be able to retire on the money. On the other hand, it will probably compensate you for the time of sitting down and typing out your ideas in a coherent fashion to share with other readers. But one way or another, as an author or as a commentator or as a critic, let us hear from you.

Over the last three years, our subscription solicitation letters have frequently invited people to join the *Creative Computing* family. I still think of the entire readership of *Creative Computing* in that light, in their many and varied roles, as being part of one large family, one large family of intelligent, wise computer users. I hope you do too. ■

David H. Ahl

$12.95

The Best of
creative computing
Volume 3

From the #1 magazine of software
and applications
comes this potpourri of information about:

Resources
Problem Solving
Public Access to Computers
Technology—Present and Future
Computers in Medicine, Science, Music, and the Arts
Programs, Simulations, Games and Puzzles
Reviews of Hardware, Software, and Books
Languages and Programming Theory
Computers in Education
Fiction

Creative Computing Press Morris Plains, New Jersey

ISBN-0-916688-12-7

Ahl & Green

The Best of Creative Computing Volume 3

Creative Computing Press